Finding structure in permutation sets

Christian Bean
Doctor of Philosophy
June 2018
School of Computer Science
Reykjavík University

Ph.D. Dissertation

Finding structure in permutation sets

by

Christian Bean

Dissertation submitted to the School of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

June 2018

The undersigned hereby certify that they recommend to the School of Computer Science at Reykjavík University for acceptance this Dissertation entitled **Finding structure in permutation sets** submitted by **Christian Bean** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy (Ph.D.) in Computer Science**

....

date

....

Henning Ulfarsson, Supervisor
Assistant Professor, Reykjavík University, Iceland

....

Anders Claesson, Co-supervisor
Professor, University of Iceland, Iceland

....

Jay Pantone, Examiner
John Wesley Young Research Instructor, Dartmouth College, USA

....

Einar Steingrímsson, Examiner
Professor, University of Strathclyde, UK

Finding structure in permutation sets

Christian Bean

June 2018

Abstract

New automatic methods for enumerating permutation classes are introduced. The first is Struct, which is an algorithm that conjectures a structural description using rules similar to generalized grid classes. These conjectured structural descriptions can be easily enumerated and are easily verified by a human to be correct.

We then introduce the CombSpecSearcher algorithm, a general framework for searching for combinatorial specifications. To use this, one must write strategies that explain how combinatorial classes are related. We introduce strategies for permutation classes that are used by the CombSpecSearcher algorithm.

We provide an algorithm for finding the *insertion encoding* of regular insertion encodable permutation classes. Our approach requires less generation of permutations and as such is much faster than previous implementations. The algorithm relies on *tilings*; a new object introduced that can be used to encode geometric proof ideas for permutation patterns efficiently.

After developing the theory of tilings briefly, we encode more geometric proof ideas that allow for placing points, each such leading to a new algorithm. All of these algorithms search for combinatorial specifications using the CombSpecSearcher which can then be enumerated. The algorithms introduced that use CombSpecSearcher and tilings we collectively call Tilescope.

We introduce the notion of an *elementary permutation class*. We show that our definition is equivalent to the permutation class being a disjoint union of *generalized peg permutations*, and as such all *polynomial* permutation classes are elementary. We show that such a description for a permutation class can be extended to permutation classes where the basis has an extra pattern.

The methods introduced in this thesis can enumerate all permutation classes with six or more length four patterns. There are only 77 bases consisting of only length four patterns which are not enumerated by our methods.

Titll verkefnis

Christian Bean

júní 2018

Útdráttur

Nýjar sjálfvirkar aðferðir til að telja umraðanaflokka eru kynntar. Sú fyrsta er Struct, sem er reiknirit til að búa til tilgátur um uppbyggingu flokka með alhæfðum grindargerðum. Þessar tilgátur er auðvelt að nota til talninga og staðfesta af manneskju.

Síðan kynnum við CombSpecSearcher, almenna umgjörð til að leita að fléttufræðilegum forskriftum. Til að nota umgjörðina þarf að skrifa kænskur sem útskýra hvernig fléttufræðilegir flokkar tengjast. Við kynnum kænskur fyrir umraðanaflokka og notum þær í umgjörðinni.

Við gefum reiknirit til að finna *innsetningarumritunina* fyrir umraðanaflokka sem hafa reglulega innsetningarumritun. Aðferð okkar þarfnast minni framleiðslu umraðana og er því hraðvirkari en fyrri útfærslur. Reikniritið byggir á *flísum*; nýs hlutar sem hægt er að umrita rúmfræðilegar sönnunaraðferðir með. Eftir að hafa þróað fræði flísa þá umritum við sönnunaraðferðir til þess að staðsetja punkta og fáum þannig ný reiknirit. Öll þessi reiknirit nýta CombSpecSearcher-umgjörðina og leiða til talninga. Við gefum þessum reikniritum yfirheitið Tilescope.

Við kynnum *grundvallarumraðanaflokka* og sýnum að það eru þeir umraðanaflokkar sem eru ósamsniða sammengi alhæfðra pinnaumraðana. Þar af leiðandi eru allir margliðuumraðanaflokkar grundvallarumraðanaflokkar. Við sýnum að viðbót á mynstri við grunn grundvallarumraðanaflokks gefur annan grundvallarumraðanaflokk.

Aðferðirnar í þessari ritgerð geta talið alla umraðanflokka með sex eða fleiri mynstur af lengd fjórum. Það eru einungis 77 grunnar með mynstur af lengd fjórum sem aðferðir okkar geta ekki talið.

In memory of my grandmother (or Nanny to me) Janet Stewart.

Perhaps I've taken our Sunday afternoon counting puzzles too far?

Acknowledgements

First, I'd like to thank Laura, whose emotional support has gotten me through this thesis and has aided in ensuring that living in a new country has been a great experience. I also extend my thanks to my family, my mum Maureen, and brothers Andrew and Stephen for supporting me throughout all my endeavours and encouraging me to reach for my goals. Thanks also go to the numerous friends I have been lucky enough to make during my studies in Iceland of which there are too many to list.

I would like to thank my supervisor Henning Ulfarsson for his unwavering enthusiasm for permutation patterns throughout my Ph.D. I would also like to thank my co-supervisor Anders Claesson for improving my ability to present mathematics. I'm also grateful to my collaborators Michael Albert who helped simplify our algorithm particularly in Section 3.3, and Jay Pantone for helping me appreciate my work and teaching me the wonders of the catalytic variable.

During my Ph.D. I have had the opportunity to work with many BSc and MSc students including Ragnar Árdal, Bjarki Guðmundsson, Tómas Ken Magnússon, Murray Tannock, Arnar Arnarson, Álfur Bjarnason, Unnar Erlendsson, Sigurður Helgason, James Robb, and Sigurjón Viktorsson. Working with these people has undoubtedly improved my ability as a computer scientist for which I thank them all.

The research in this thesis was partially supported by grant 141761-051 from the Icelandic Research Fund.

Contents

Acknowledgements **xiii**

Contents **xv**

List of Figures **xvii**

List of Tables **xix**

1 Introduction **1**
- 1.1 Combinatorics . . . 1
- 1.2 Permutation patterns . . . 2

2 Geometric view of permutations **5**
- 2.1 Classical permutation patterns . . . 5
- 2.2 Bivincular patterns . . . 6
- 2.3 The Struct algorithm . . . 9
- 2.4 Polynomial permutation classes . . . 13
- 2.5 Enumerating Struct covers . . . 14
- 2.6 Results of the Struct algorithm . . . 16

3 Combinatorial exploration **19**
- 3.1 Combinatorial classes . . . 19
- 3.2 Combinatorial expansion . . . 20
- 3.3 Searching for a combinatorial specification . . . 22

4 Gridded permutations **25**
- 4.1 Figures and tilings . . . 25
- 4.2 Minimal obstructions and requirements . . . 28
- 4.3 Counting tilings . . . 31

5 Revisiting the insertion encoding **35**
- 5.1 Insertion encoding . . . 35
- 5.2 Row separation . . . 38
- 5.3 Allowing more directions . . . 40
- 5.4 Successes . . . 42

6 Point placement **45**
- 6.1 Placing points on a tiling . . . 45
- 6.2 Stretching gridded permutations . . . 46
- 6.3 Forced points in requirements . . . 47

6.4 Requirement insertion and placement 49
6.5 Elementary combinatorial specifications 51
6.6 Successes 55
6.7 Combinatorial systems 57

7 Conclusion **63**
7.1 Permutations avoiding 123 63
7.1.1 Left-to-right minima 63
7.1.2 Interleaving factors 65
7.2 Other combinatorial classes 66
7.3 Open questions 66

Bibliography **69**

List of Figures

2.1 The diagram of the permutation $\pi = 68153724$. The circled points show an occurrence of 1342. 5
2.2 The diagram of the permutation $\pi = 21843756 \in \mathrm{Av}(231)$. The circled entry highlights the topmost point. Here $\alpha = 21$ and $\beta = 43756$. 6
2.3 A graphical depiction of the structure of an arbitrary non-empty permutation in Av(231). Here, $C = \mathrm{Av}(231)$. 6
2.4 $(231, \emptyset, \emptyset)$, $(123, \{2\}, \emptyset)$, $(123, \{1\}, \emptyset)$, $(312, \emptyset, \{2\})$, and $(312, \emptyset, \{1\})$. 7
2.5 An example Struct rule on the left, alongside its factors on the right. 14

4.1 A drawing of the figure $\mathcal{F}$ from Equation (4.1) that is in the equivalence class represented by $2^{(0,0)}8^{(0,3)}4^{(1,1)}3^{(1,1)}7^{(2,3)}6^{(2,2)}9^{(3,4)}1^{(3,0)}5^{(4,2)}$. 26
4.2 The left hand picture is a representation of the tiling $\mathcal{T}$ in Equation (4.2), called the point tiling. As the set $\mathrm{Grid}(\mathcal{T})$ contains only the gridded permutation $1^{(0,0)}$ we draw it as a point as in the right hand side tiling. 28
4.3 The left hand tiling $\mathcal{T}_1$ has the gridded permutations $\pi_1 = 1^{(0,0)}3^{(0,0)}2^{(1,0)}4^{(1,1)}$ and $\pi_2 = 1^{(0,0)}2^{(0,0)}3^{(1,0)}4^{(1,1)}$ as the requirement $\mathcal{R}_1 = \{\pi_1, \pi_2\}$. The right hand tiling $\mathcal{T}_2$ has the maximal elements of intersection of π_1 and π_2 as two size one requirements. By Proposition 4.2.1 we know $\mathrm{Grid}(\mathcal{T}_1) \subseteq \mathrm{Grid}(\mathcal{T}_2)$. 29
4.4 The left tiling $\mathcal{T}_1$ has the requirement $\left\{\pi = 1^{(0,0)}2^{(0,0)}5^{(1,2)}6^{(1,2)}3^{(2,1)}4^{(3,2)}7^{(3,3)}\right\}$, and the right tiling $\mathcal{T}_2$ has a requirement for each factor in $\mathbb{F}(\pi)$. By Lemma 4.2.2, we know $\mathrm{Grid}(\mathcal{T}_1) = \mathrm{Grid}(\mathcal{T}_2)$. 30
4.5 Two tilings that have the same set of gridded permutations but have two different minimal sets of obstructions with respect to set containment. 31
4.6 A tiling and its factors. 32
4.7 The Struct cover from Proposition 2.3.2 in terms of tilings. 33

5.1 The evolution of the permutation 325146. 35
5.2 The tilings for the prefixes ϵ, $\mathbf{f}_1$, $\mathbf{l}_1$, $\mathbf{r}_1$, and $\mathbf{m}_1$, respectively, as described in Lemma 5.1.2 for Av(231), where for each we have the tiling using the methods of Section 4.2 and removing rows and columns that are empty. 37
5.3 A pictorial representation of the combinatorial specification for Av(312) found by Tilescope using the separation insertion encoding. 39
5.4 A pictorial representation of the combinatorial specification found by Tilescope for Av(2143, 2413, 3142, 3412) using top and bottom insertion encoding. 41

6.1 On the left is a tiling $\mathcal{T}$ with a point requirement at cell (2, 2). On the right is the tiling $\mathcal{T}^{(6,6)}_{(2,2)}$ with the point requirement of the left tiling placed on a new row and column. The regions enclosed by the bold lines correspond to the cells of $\mathcal{T}$. In Sections 6.2 and 6.3 we discuss how to make this operation a combinatorial rule. 45

6.2 A gridded permutation in red, with black circles representing where a placed point in the region could be with respect to the gridded permutation. 46
6.3 A pictorial representation of the combinatorial specification found by Tilescope for Av(1243, 1342, 2143). 52
6.4 A pictorial representation of the combinatorial system found by Tilescope for Av(1243, 1234, 1324, 1423, 2134, 2314). 60

7.1 An avoider of 123 with five left-to-right minima where A contains at least one point 63
7.2 The sum of two 123-avoiding permutations 64
7.3 The sum of the empty permutation and a 123-avoiding permutation 64

List of Tables

2.1 Data from running Struct on every non-polynomial basis with length four patterns. 16

5.1 The successes for bases consisting of length 4 patterns with insertion encoding strategies. 42

6.1 The successes for bases consisting of length 4 patterns with elementary point placement strategies. 55
6.2 The successes for bases consisting of length 4 patterns with point placement strategies. 56
6.3 Bases consisting of two length 4 patterns that succeed with point placement strategies. They are listed with the number of equations the combinatorial specification has, their OEIS sequence, and the reference to the paper that first enumerated it. 56
6.4 The successes for bases consisting of length 4 patterns with combinatorial systems using point placement strategies. 61

Chapter 1

Introduction

1.1 Combinatorics

Counting is the act of determining the number of objects there are in a set. To enumerate a set, you assign each object a number. You might do this by pointing at the first object and labeling it one, the next object two, and so on ensuring to label every object exactly once. The last label you assign is then the number of objects in your set. This method is a basic case of a bijection and a useful technique for counting finite sets.

Combinatorics is often described as the mathematics of counting. To be precise, *enumerative combinatorics* is the study of counting finite structures. Our finite structures tend to have some notion of size, and the question becomes how many of each size are there. What constitutes a good answer to this question can get philisophical[1] but could, for example, be a formula such as $a_n = \frac{n(n-1)}{2}$ which counts the number of ways of choosing two distinct numbers from the set $\{1, 2, \ldots, n\}$. The formula $a_n = \frac{n(n-1)}{2}$ determines the answer for all n and gives a sequence of numbers

$$0, 0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, \ldots.$$

Another method is to give a bijection to a set already counted. For example, this sequence also counts the number of ways to pick two numbers with replacement from the set $\{1, 2, \ldots, n-1\}$. If (a, b) is an ordered such pair, so $a \leq b$, then add 1 to b. This pair will be two distinct numbers from the set $\{1, 2, \ldots, n\}$. Similarly, if (a, b) are two distinct numbers with $a < b$ from the set $\{1, 2, \ldots, n\}$ then subtract 1 from b and you will have a pair of numbers from $\{1, 2, \ldots, n-1\}$ that are possibly equal. We have described a bijection and, therefore, these two different things are counted the same.

A sequence of numbers $(a_n)_{n\geq 0}$ can be encoded in a *generating function* described by Wilf [58] as "a clothesline on which we hang up a sequence of numbers for display". A generating function is a power series whose coefficients are the numbers in your sequence.

$$\sum_{n\geq 0} a_n x^n$$

A generating function often has a closed form in terms of the indeterminate x which looks like a function of x, for example

$$\sum_{n\geq 0} \frac{n(n-1)}{2} x^n = \frac{x^2}{(1-x)^3},$$

[1]see for example Wilf [59] for an article discussing this.

which has the power series as its series expansion. This is a compact way to answer this question.

By describing how to construct a finite structure, the *symbolic method*, detailed in Flajolet and Sedgewick [30], often allows you to write down an equation or system of equations satisfied by the generating function. The equations can then be solved to give the generating function and thus answering the question. This particular approach is the one that will be used predominantly throughout this thesis. In Chapter 2 we will get some practice using the symbolic method and in Chapter 3 we will define it more carefully as we aim to automate research with this method.

1.2 Permutation patterns

The finite structures of interest are *permutations* and, in particular, *permutation patterns*. Precise definitions are deferred to Chapter 2. The study of permutation patterns can be dated back to 1915 when MacMahon [43] showed that the number of permutations of length n that can be partitioned into two decreasing subsequences is given by the n^{th} Catalan number. These permutations can be equivalently defined as those which *avoid* the (*classical*) permutation pattern 123. The field took off due to an exercise in Knuth [38, p. 243, ex. 5] that asked you to show that a permutation is stack-sortable if and only if it avoids 231. The Catalan numbers also count these permutations. Permutation sets defined by avoidance of permutation patterns are called *permutation classes*.

There has been a movement towards a generalized approach for enumerating permutation classes.

(1) One of the first completely automatic approaches was a method to find *enumeration schemes* introduced by Zeilberger [60] and considerably extended by Vatter [54]. The goal of enumeration schemes is to break up a permutation class into smaller parts and find recurrence relations. There is no general theory for when a permutation class has a finite enumeration scheme, but when it does succeed this is a polynomial time algorithm.

(2) The *insertion encoding* is an encoding of finite permutations, introduced by Albert, Linton, and Ruškuc [6], and generalizes the finitely labeled generating trees of Chung *et al.* [24]. It encodes how a permutation is built up by iteratively adding a new maximum element. In particular, Albert, Linton, and Ruškuc [6] studied the permutation classes whose insertion encodings are regular languages, including giving a characterization of these permutation classes. For regular insertion encodable permutation classes, Vatter [55] provides an algorithm for automatically computing the rational generating function. We will revisit this in more detail in Chapter 5.

(3) Albert and Atkinson [1] were the first to apply the substitution decomposition to the enumeration of permutation classes. In this approach, one views permutations as inflations of simple permutations. There are automatic methods for enumerating permutation classes with the substitution decomposition when the class under inspection contains finitely many simple permutations. For wreath-closed permutation classes Bassino *et al.* [13] gave an $O(n \log n)$ algorithm for determining if the permutation class contains finitely many simple permutations. This algorithm was extended to an $O(n \log n + s^{2k})$ algorithm for general permutation classes by Bassino *et al.* [12].[2] The entire procedure

[2]In these algorithms n denotes the sum of the lengths of the patterns the permutations avoid, and s and k depend on specific properties of the permutation class.

for enumerating permutation classes with finitely many simple permutations has been implemented by Bassino et al. [11].

(4) In the case of polynomial permutation classes[3], it was shown by Homberger and Vatter [33], by combining results by Albert *et al.* [7] and Huczynska and Vatter [34], that such a permutation class can be represented by a finite set of peg permutations. From this finite set of peg permutations, Homberger and Vatter [33] give an automatic method to enumerate polynomial permutation classes. However, it is not known in general how to find the set of peg permutations.

In this thesis, a few different algorithms are introduced for enumerating permutation classes. The Struct algorithm is discussed in Section 2.3. It takes as input a set of permutations or a permutation property and conjectures certain structural rules that can lead to a generating function enumerating the input. When the algorithms mentioned in (1)-(4) succeed, they output a proof, whereas Struct only outputs conjectures. However, the produced conjectures usually turn out to be easily verified by a human.

The CombSpecSearcher algorithm, introduced in Chapter 3, can find combinatorial specifications automatically, which in turn proves the enumeration of combinatorial classes. To use this, you must provide CombSpecSearcher with sets of strategies that explain how combinatorial classes are related. As such, it represents multiple algorithms, as each set of strategies gives a different algorithm.

In Chapter 5, we give an alternative implementation for finding the insertion encoding of regular insertion encodable classes that uses the CombSpecSearcher algorithm. We extend this in a natural way which leads to the enumeration of more permutation classes. Finally, in Chapter 6, we introduce our most powerful algorithm that also uses the CombSpecSearcher algorithm.

[3]A permutation class is said to be *polynomial* if the number of length n permutations in the permutation class is given by a polynomial for all sufficiently large n.

Chapter 2

Geometric view of permutations

2.1 Classical permutation patterns

A length n *permutation* is an ordering of the integers $\{1, 2, \ldots, n\}$. The set of length n permutation is denoted $\mathcal{S}_n$. For example, $\mathcal{S}_3 = \{123, 132, 213, 231, 312, 321\}$, where the permutations have been written in *one-line notation*. The set of all permutations is denoted $\mathcal{S}$.

Define the *standardization* of a string, $s_1 s_2 \cdots s_k$, of distinct numbers as the permutation created by replacing the i^{th} smallest entry with i. For example, the standardization of 2673 is 1342.

A permutation π *contains* σ, written $\sigma \leq \pi$, if there exist indices $i_1 < i_2 < \cdots < i_k$ such that the standardization of the subsequence $\pi_{i_1}\pi_{i_2} \cdots \pi_{i_k}$ is σ. In this context, σ is called a *(classical) permutation pattern*. Each set of indices witnessing the containment of σ is called an *occurrence* of σ in π. If π does not contain σ then π *avoids* σ.

For example, the permutation $\pi = 68153724$ contains the pattern 1342. This can be seen by the occurrence $\{3, 4, 6, 8\}$ which corresponds to the subsequence 1574. It has two more occurrences of 1342 at indices $\{3, 4, 6, 7\}$ and $\{3, 5, 6, 7\}$.

The *diagram* of a permutation π is the plot of points (i, π_i), on the Cartesian plane. In Figure 2.1 is an example of the diagram of $\pi = 68153724$ with the circled entries showing an occurrence of 1342 at the indices $\{3, 4, 6, 8\}$.

The containment relation forms a partial order on the set of permutations. Any set of permutations that is closed downwards with respect to the permutation containment order ($\leq$) is called a *permutation class*. Another way to define a permutation class is by the minimal (possibly infinite) set of permutations not in the class, called the *basis*. A permutation π

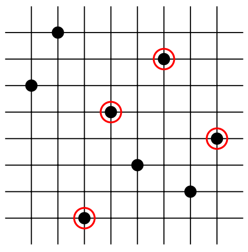


Figure 2.1: The diagram of the permutation $\pi = 68153724$. The circled points show an occurrence of 1342.

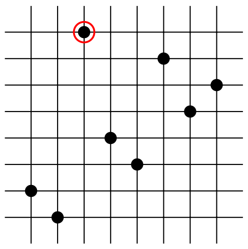


Figure 2.2: The diagram of the permutation $\pi = 21843756 \in \mathrm{Av}(231)$. The circled entry highlights the topmost point. Here $\alpha = 21$ and $\beta = 43756$.

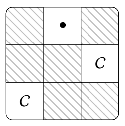


Figure 2.3: A graphical depiction of the structure of an arbitrary non-empty permutation in Av(231). Here, $C = \mathrm{Av}(231)$.

avoids a set of patterns Π if it avoids every permutation in Π. Let

$$\mathrm{Av}(\Pi) = \{\pi \in \mathcal{S} \mid \pi \text{ avoids } \Pi\}$$

be the permutation class with basis Π, and $\mathrm{Av}_n(\Pi)$ be the length n permutations in this permutation class. The *containers* of a pattern σ is

$$\mathrm{Co}(\sigma) = \{\pi \in \mathcal{S} \mid \pi \text{ contains } \sigma\} = \mathcal{S} \backslash \mathrm{Av}(\sigma)$$

and $\mathrm{Co}_n(\sigma)$ is the set of length n permutations in this set.

The question we focus our attention on is, given a permutation class $\mathrm{Av}(\Pi)$, can we determine the sequence $(|\mathrm{Av}_n(\Pi)|)_{n\geq 0}$? An answer to this can be in the form of a closed formula or a generating function for the sequence, see for example Wilf [59] for discussion of what is considered a satisfactory answer. When we get an answer to this question we say we have *enumerated* the permutation class, and call the answer its *enumeration*. Two permutation classes with the same enumeration are called *Wilf-equivalent*.

As an example, let us enumerate Av(231). Every permutation either has no points or a topmost point. That is, for $n \geq 1$ every length n permutation can be written as $\pi = \alpha n \beta$. If π avoids 231 then all the points in α must be below all the points in β, else there would be a 231 pattern in π. Moreover, if π avoids 231 then both α and β must avoid 231, for an example see Figure 2.2. This describes the structure of the permutations in Av(231). From the structure one can obtain the enumeration with the formula (where $c_n = |\mathrm{Av}_n(231)|$, and in particular $c_0 = 1$)

$$c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} = \frac{1}{n+1}\binom{2n}{n}$$

for the n^{th} Catalan number. The argument of the structure can be viewed pictorially as in Figure 2.3. Such a structure is what is searched for by the Struct algorithm in Section 2.3.

2.2 Bivincular patterns

Babson and Steingrímsson [10] introduced a generalization of classical patterns that allow the requirement that two adjacent letters in a pattern must be adjacent in the permutation. These are called *vincular patterns*. A further extension, called *bivincular patterns* which also added requirements that consecutive values of a pattern must be consecutive in value in the permutation, was provided by Bousquet-Mélou *et al.* [21]. We call the special case when only constraints on values are allowed *covincular patterns*. The set of bivincular patterns is closed under the action of the symmetry group of the square and an alternative way of describing the covincular patterns is that they are inverses of vincular patterns. In the following definition, the sets X and Y are for the vincular and covincular restrictions respectively.

Definition 2.2.1 (Bousquet-Mélou *et al.* [21, page 4]). A length k *bivincular pattern* is a triple, $p = (\sigma, X, Y)$, where $\sigma \in \mathcal{S}_k$ is the *underlying permutation* and X and Y are subsets of $\{0, 1, \dots, k\}$. An *occurrence* of p in $\pi \in \mathcal{S}_n$ is a subsequence $w = \pi_{i_1} \cdots \pi_{i_k}$ such that the standardization of w is σ and

$$\forall x \in X,\ i_{x+1} = i_x + 1 \quad \text{and} \quad \forall y \in Y,\ j_{y+1} = j_y + 1,$$

where $\{\pi_{i_1}, \dots, \pi_{i_k}\} = \{j_1, \dots, j_k\}$ and $j_1 < j_2 < \cdots < j_k$. By convention, $i_0 = j_0 = 0$ and $i_{k+1} = j_{k+1} = n + 1$. If such an occurrence exists we say that π *contains* p.

Further, a permutation *avoids* p if it does not contain p. If $Y = \emptyset$ then p is a *vincular* pattern. If $X = \emptyset$ then p is a *covincular* pattern. If $X = Y = \emptyset$ then p is a *classical* pattern. For example, the permutation 15423 contains an occurrence of $(123, \{2\}, \emptyset)$, namely the subsequence 123, but avoids $(123, \{1\}, \emptyset)$. The permutation 23514 contains an occurrence of $(312, \emptyset, \{2\})$, namely the subsequence 523, but avoids $(312, \emptyset, \{1\})$.

Below we use a pictorial representation of vincular and covincular patterns. For a length n bivincular pattern $p = (\sigma, X, Y)$: First draw the collection of points from the underlying permutation i.e. (k, σ_k) where $1 \leq k \leq n$. Then, for each $i \in X$, shade the ith column and, for each $j \in Y$, shade the jth row; see Figure 2.4. The shading is used to denote the empty regions in the permutation if we were to overlay the grid onto an occurrence of the pattern.

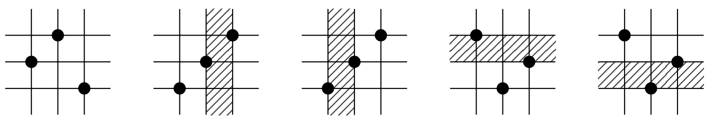
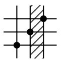
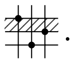


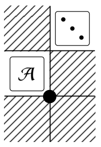
Figure 2.4: $(231, \emptyset, \emptyset)$, $(123, \{2\}, \emptyset)$, $(123, \{1\}, \emptyset)$, $(312, \emptyset, \{2\})$, and $(312, \emptyset, \{1\})$.

Simultaneous avoidance of two vincular patterns was studied by Claesson and Mansour [26] and by Kitaev [37]. Allowing one of the patterns to be covincular is a natural follow up question and leads to some well-known sequences. In Bean, Claesson, and Ulfarsson [15] the goal was to count the number of permutations simultaneously avoiding a length 3 vincular and a length 3 covincular pattern, where both patterns force at most one restriction. We will give some of the results from there which will be used to illustrate the automatic methods of Section 2.3.

Proposition 2.2.2. Let $p =$  and $r =$ . The number of permutations in

$$\mathrm{Av}_n(p, r) \quad \text{is} \quad 2^{n-1}, \quad \text{for} \quad n \geq 1.$$

Proof. Let $\mathcal{A}$ be the set of avoiders in question and let $\pi \in \mathcal{A}$. As π avoids p, the points after the minimum of π form a decreasing sequence. Moreover, in order to ensure that the permutation avoids r, every point to the right of the minimum must be greater than every point on the left of the minimum. Therefore, all non-empty permutations of $\mathcal{A}$ have the form

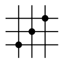
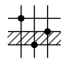


where $\mathcal{A}$ symbolizes a (possibly empty) permutation which avoids the patterns, and $\ddots$ symbolizes a decreasing permutation. As the structure is so rigid we can find the ordinary generating function of the avoiders by multiplying together the ordinary generating functions of the component parts. There is one decreasing permutation of length n and so the ordinary generating function is $1/(1-x)$. The ordinary generating function of a single point is x. If A is the ordinary generating function for $\mathcal{A}$, then it follows that

$$A = A \cdot x \cdot \frac{1}{1-x} + 1$$

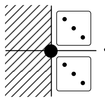
where we add 1 for the empty permutation which trivially avoids both patterns. Rearranging we get

$$A = \frac{1-x}{1-2x} = 1 + \sum_{n\geq 1} 2^{n-1}x^n. \qquad \square$$

Proposition 2.2.3. Let $p =$  and $r =$ . The number of permutations in

$$\mathrm{Av}_n(p,r) \quad \text{is} \quad 2^{n-1}, \quad \text{for} \quad n \geq 1.$$

Proof. Let $\mathcal{A}$ be the set of avoiders in question. Consider the leftmost point ℓ of a permutation in $\mathcal{A}$. To avoid p the points greater than ℓ must form a decreasing sequence and similarly to avoid r the points less than ℓ must form a decreasing sequence. Therefore the non-empty permutations in $\mathcal{A}$ have the form

.

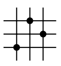
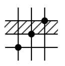
A permutation matching this picture cannot contain an occurrence of $p = 123$, and every occurrence of 312 will have the point ℓ preventing it from being an occurrence of r. Hence these can be encoded with binary strings and so there are 2^{n-1} such permutations. Alternatively, if we let A be the exponential generating function for $\mathcal{A}$ then

$$A = 1 + \int e^x \cdot e^x dx = 1 + \frac{e^{2x}}{2} = 1 + \sum_{n\geq 1} \frac{2^{n-1}x^n}{n!}. \qquad \square$$

The Motzkin numbers, M_n, form a well-known sequence which can be defined by a functional equation their ordinary generating function satisfies:

$$M = 1 + xM + x^2M^2 \quad \text{where} \quad M = \sum_{n\geq 0} M_n x^n.$$

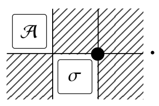
For more information on Motzkin numbers see e.g., sequence A001006 on the OEIS, Sloane [52], and Donaghey and Shapiro [28].

Proposition 2.2.4 (Elizalde and Mansour [29])**.** Let $p =$  and $r =$ . The number of permutations in

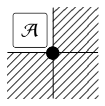
$$\mathrm{Av}_n(p, r) \quad \text{is} \quad M_n.$$

The proof given by Elizalde and Mansour [29] provides a bijection between $\mathrm{Av}_n(p, r)$ and Motzkin paths. We show that the structure of the permutations implies they are enumerated by the Motzkin numbers.

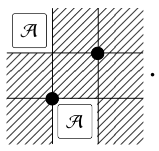
Proof. Consider a permutation π in $\mathcal{A} = \mathrm{Av}(p, r)$. Further, consider the rightmost point of π. For π to avoid p the structure of π must be like



With regard to σ, let us consider two cases. Either σ is empty or it has at least one point. If σ is empty the structure looks like

 (2.1)

If σ is non-empty then consider the maximum point, m, of σ. If there was a point to the left of m in σ then this point together with m and the rightmost point of π would create an occurrence of r. Therefore there must be no points to the left of m in σ. Thus we can place any possibly empty smaller permutation in $\mathcal{A}$ to the right of the maximum of σ without creating an occurrence of p or r, and so we have the structure

 (2.2)

In conclusion, any non-empty permutation in $\mathcal{A}$ either has a structure described by (2.1) or a structure described by (2.2). Letting A denote the ordinary generating function for $\mathcal{A}$ we thus have $A = 1 + xA + x^2A^2$, from which the claim follows. □

2.3 The Struct algorithm

In Bean, Gudmundsson, and Ulfarsson [16] we introduced Struct, an algorithm which takes as input a set of permutations or a permutation property and conjectures certain structural rules that can lead to a generating function enumerating the input. We emphasize that Struct outputs conjectures but note that the produced conjectures usually turn out to be easily verified by a human. In Chapter 3 and Chapter 4 we discuss an algorithm for finding proofs rather than conjectures.

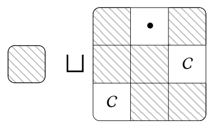
We first review the generalized grid classes introduced by Vatter [56]. For non-negative integers n and m, define the intervals $[n] = \{1, 2, \ldots, n\}$, $[m, n] = \{m, m + 1, \ldots, n\}$ and $[m, n) = \{m, m+1, \ldots, n-1\}$. Given a permutation π of length n, and two subsets $X, Y \subseteq [n]$, then $\pi(X \times Y)$ is the permutation that is order isomorphic to the subword with indices from X and values in Y. For example $35216748([3, 7] \times [2, 6]) = 132$, from the subword 264.

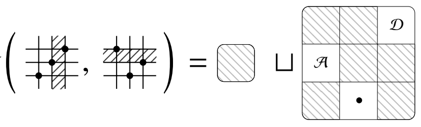
Suppose M is a $t \times u$ matrix (indexed from left to right and bottom to top) whose entries are permutation sets[1]. An *M-gridding* of a permutation π of length n is a pair of sequences $1 = c_1 \leq \cdots \leq c_{t+1} = n+1$ and $1 = r_1 \leq \cdots \leq r_{u+1} = n+1$ such that $\pi([c_k, c_{k+1}) \times [r_l, r_{l+1}))$ is in $M_{k,l}$ for all k in $[t]$ and ℓ in $[u]$. The *generalized grid class* of M, Grid(M), consists of all permutations with an M-gridding.

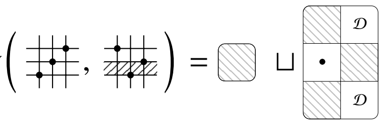
A (*Struct*) *rule* $\mathcal{R}$ is a matrix whose entries are permutation sets, with the requirement that each permutation π in the grid class Grid($\mathcal{R}$) has a unique $\mathcal{R}$-gridding. From now on we will abuse notation and use $\mathcal{R}$ to denote both the Struct rule and its grid class. Let $\mathcal{R}_{\leq n}$ be the set of permutations in $\mathcal{R}$ of length at most n.

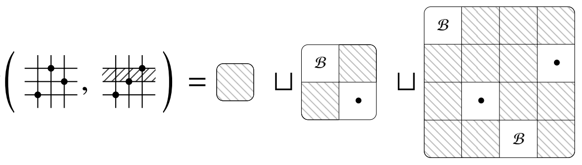
For a permutation set S, Struct tries to write it as a disjoint union of rules. We call this a Struct *cover*. We can restate the propositions from Section 2.2 in this language.

Proposition 2.3.1 (Propositions 2.2.2, 2.2.3 and 2.2.4)**.** The following are Struct covers, where $\mathcal{D} = \text{Av}(12)$.

1. $\mathcal{C} = \text{Av}(231) =$ 

2. $\mathcal{A} = \text{Av}$ 

3. Av 

4. $\mathcal{B} = \text{Av}$ 

The Struct algorithm consists of four main steps for finding a cover for a permutation set S. The input is S and an integer ℓ being the maximum size of the rules considered.

1. Generate the *block set*, that is the set of permutation sets used in the rules.

2. Generate Struct rules[2], $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_K$ up to $\ell \times \ell$ with entries from the block set, satisfying $\mathcal{R}_{\leq \ell+1} \subseteq S$. The trivial rule $\boxed{S}$ is discarded.

3. Try to find a cover of $S_{\leq \ell+1} = \{\pi \in S \mid |\pi| \leq \ell + 1\}$ with the rules from the previous step, i.e., write this set as a disjoint union

$$S_{\leq \ell+1} = \mathcal{R}_{i_1, \leq \ell+1} \sqcup \cdots \sqcup \mathcal{R}_{i_k, \leq \ell+1}.$$

4. If a cover is found, verify that it remains valid in length $\leq \ell + 3$, i.e.,

$$S_{\leq \ell+3} = \mathcal{R}_{i_1, \leq \ell+3} \sqcup \cdots \sqcup \mathcal{R}_{i_k, \leq \ell+3}.$$

[1]In the original definition given by Vatter [56] the entries are permutation classes

[2]Candidates for Struct rules are experimentally checked to not create duplicates up to length $\ell + 1$.

The bounds mentioned are somewhat arbitrary and can be increased by the user. By providing different settings, the user can make the algorithm look for a cover with larger rules, or verify a found cover for longer permutations. In step (3) we use the integer linear programming solver Gurobi (Gurobi Optimization, Inc. [31]) to find a minimal cover if one exists; resorting to the SAT solver Lingeling (Biere [19]) if Gurobi runs out of memory.

In Bean, Gudmundsson, and Ulfarsson [16] we focussed on permutation classes and assumed a finite basis, Π, was given. The blocks of our rules are the non-finite subclasses of the permutation class that are formed by taking subpatterns of the elements in the basis of the permutation class. More formally, define the set $\Delta(\pi)$ to be the set of all patterns contained in π,

$$\Delta(\pi) = \{\sigma \in \mathcal{S} : \sigma \preceq \pi\}.$$

For a set of patterns Π, let $\Delta(\Pi) = \bigcup_{\sigma\in\Pi} \Delta(\sigma)$. A permutation class could also be defined as a set C such that $\Delta(C) = C$.

A *block* of a permutation class $C = \operatorname{Av}(\Pi)$ is a permutation class $C' = \operatorname{Av}(\Pi')$, containing infinitely many permutations, such that $\Pi' \subseteq \Delta(\Pi)$ and $\Pi' \cap \Delta(\sigma) \neq \emptyset$ for all σ in Π. We also allow the finite permutation class $\operatorname{Av}(1) = \{\epsilon\}$ as a block for any permutation class C. Additionally, if $1 \in C$ we allow $\{1\}$ as a block, even though this is not a permutation class, and contains only one permutation. We call 1 "the point" and denote it with $\bullet$.

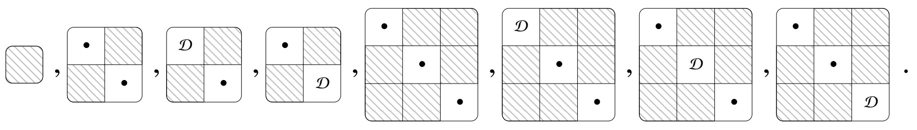
The *block set*, $\mathbb{M}(\Pi)$, of $C = \operatorname{Av}(\Pi)$ is the set of all blocks of C. We note that for a finite basis the block set of the permutation class $\operatorname{Av}(\Pi)$ is always finite. For example,

$$\mathbb{M}(\{231\}) = \{\operatorname{Av}(1), \{\bullet\}, \operatorname{Av}(12), \operatorname{Av}(21), \operatorname{Av}(231)\}.$$

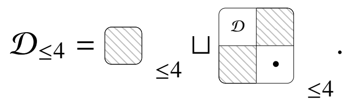
Note that $\operatorname{Av}(12, 21)$ is not a block since it is finite. As another example we have

$$\mathbb{M}(\{231, 1234\}) = \{\operatorname{Av}(1), \{\bullet\}, \operatorname{Av}(12), \operatorname{Av}(123, 231), \operatorname{Av}(231, 1234)\}.$$

As a simple example consider the decreasing permutations $\mathcal{D} = \operatorname{Av}(12)$, where $\ell = 3$. Step (1) finds the block set $\mathbb{M}(\{12\}) = \{\operatorname{Av}(1), \{\bullet\}, \operatorname{Av}(12)\}$. Step (2) generates the rules



One of the covers found in step (3) is



Step (4) verifies this cover up to length 6. At this stage, a human must step in and verify that the cover remains valid for all lengths. In this case, it is obvious, and if $D(x)$ is the generating function then we can see from the cover that $D(x) = 1 + xD(x)$, so $D(x) = \frac{1}{1-x}$.

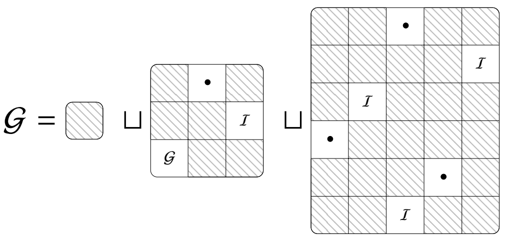
If the block set is large, then the space of possible rules to consider in step (2) will be too large to search for valid rules exhaustively. To prune the search space, we first check which blocks can share a row, column or diagonal without creating a pattern from the basis. This information is used to build the candidate Struct rules recursively without considering every possibility. We also arrange the blocks in a poset where the relation is set containment. When creating the Struct rules; if a rule with a block A in a particular cell produces the same permutation twice or a permutation outside of $\operatorname{Av}(\Pi)$ then replacing A with a block $A' \supset A$ will also not work.

The implementation of Struct can be found on GitHub, Bean, Gudmundsson, and Ulfarsson [17]. The conjectures discovered can be found on the Permutation Pattern Avoidance Library (or PermPAL for short), Arnarson *et al.* [8], alongside the conjectured enumerations.

In Bean, Gudmundsson, and Ulfarsson [16] there are many examples of Struct covers. We will look at one example to illustrate how it can be verified to be true for all lengths.

Proposition 2.3.2 (West [57], Bean, Gudmundsson, and Ulfarsson [16])**.** The structure of the permutation class $\mathcal{G} = \text{Av}(321, 1342)$ is given by the cover in Equation (2.3). The enumeration is given by A116702 and the generating function is

$$G(x) = \frac{1 - 4x + 6x^2 - 3x^3 + x^4}{1 - 5x + 9x^2 - 7x^3 + 2x^4} = 1 + x + 2x^2 + 5x^3 + 13x^4 + 32x^5 + 74x^6 + 163x^7 + 347x^8 + 722x^9 + 1480x^{10} + \cdots .$$



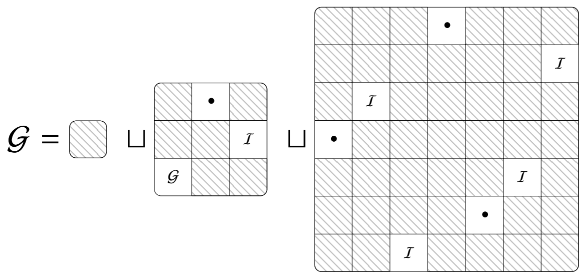
(2.3)

Proof. We will prove the equivalent cover in Equation (2.4). Let π be a permutation from $\mathcal{G}$. Either π is empty or it can be written as $\alpha n \beta$ where n is the length of π, $\alpha \in \mathcal{G}$ and $\beta \in \mathcal{I} = \text{Av}(21)$. If all the elements of α have value smaller than all the elements of β then it is not possible to create an occurrence of 321 or 1342 across α and β.

Otherwise, there exists some occurrence of 21 across α and β. Choose the occurrence cb where the points are as far to the left as possible in π. If the first element of π has value a less than c then $acnb$ would form an occurrence of 1342. If the first element to the left of n has value d greater than b then ndb would form an occurrence of 321. Therefore we can write π as $c\gamma nb\delta$ where $\gamma \in \mathcal{G}$ and $\delta \in \mathcal{I}$.

The elements of γ with value less than c must all have a value less than b to avoid 321 and, moreover, must be to the right of the elements of γ with a value greater than c to avoid 1342. Similarly to avoid 321 the elements with a value less than c in δ must be greater than b and to the left of the elements with a value greater than c.

Therefore we can write $\pi = c\gamma_1\gamma_2 nb\delta_1\delta_2$ where $\gamma_1, \gamma_2, \delta_1, \delta_2 \in \mathcal{I}$. The elements in γ_1 must have values less than the elements in δ_2 in order to avoid 1342. It is not possible to create an occurrence of 321 or 1342 across $\gamma_1, \gamma_2, \delta_1, \delta_2$. Therefore $\mathcal{G}$ has the cover



(2.4)

which is equivalent to the cover in Equation (2.3). From the cover, we obtain $G(x) = 1 + xG(x)I(x) + x^3 I(x)^4$, where $I(x) = \frac{1}{1-x}$. Solving produces the claimed equation. □

2.4 Polynomial permutation classes

A permutation class C is said to be *polynomial* if the number of length n permutations, $|C_n|$, is given by a polynomial for all sufficiently large n. One of the first general results on permutation classes by Kaiser and Klazar [36] states that if the number of length n permutations in a permutation class is less than the nth Fibonacci number for some n then the permutation class is polynomial. This is known as the *Fibonacci dichotomy* and alternative proofs were given by Huczynska and Vatter [34] and Albert, Atkinson, and Brignall [2]. From the results of Homberger and Vatter [33], we get the following theorem.

Theorem 2.4.1. All polynomial permutation classes have a cover.

In order to prove this, we will recall some definitions used by Homberger and Vatter [33]. A *peg permutation* is a permutation where each letter is decorated with a $+$, $-$ or $\circ$, for example, $\rho = 3^\circ 1^- 4^\circ 2^+$. Let M_ρ be the matrix defined by

$$M_{i,j} = \begin{cases} \mathrm{Av}(12) & \text{if } \rho_i = j^+ \\ \mathrm{Av}(21) & \text{if } \rho_i = j^- \\ \{1\} & \text{if } \rho_i = j^\circ \\ \emptyset & \text{otherwise} \end{cases}$$

then we let $\mathrm{Grid}(\rho) = \mathrm{Grid}\left(\mathrm{M}_\rho\right)$.

A peg permutation is, therefore, a generalized grid class with monotone intervals for its matrix entries. We can specify these intervals with vectors of non-negative integers. We call this a *ρ-partition*. For example, we could write

$$6321745 = 3^\circ 1^- 4^\circ 2^+[\langle 1,3,1,2\rangle].$$

Throughout it is insisted that we use vectors that *fill* peg permutations, meaning that a component of a vector equals 1 if it corresponds to a $\circ$ and otherwise is at least 2. For a set of filling vectors $\mathcal{V}$, define

$$\rho[\mathcal{V}] = \{\rho[v] : v \in \mathcal{V}\}.$$

Given vectors v and w in $\mathbb{N}^m$, then v is contained in w if $v(i) \le w(i)$ for all indices i. This is a partial order and moreover, if v is contained in w then for a length m peg permutation, $\rho[v]$ is contained in $\rho[w]$. A set closed downwards under containment is called a *downset*, and closed upwards an *upset*. The intersection of a downset and an upset is called a *convex set*.

The set of vectors which fill a given peg permutation ρ forms a convex set. The downset component of this convex set consists of those vectors which do not contain an entry larger than one corresponding to a dotted entry of ρ. The upset component consists of those vectors which contain the vector v defined by $v(i) = 1$ if $\rho(i)$ is dotted and $\rho(i) = 2$ if $\rho(i)$ is signed. As we discussed in Section 1.2, all polynomial permutation classes can be represented by a finite set of peg permutations. In fact, a more general condition holds.

We can now state the result from Homberger and Vatter [33, Theorem 1.4].

Theorem 2.4.2 (Homberger and Vatter [33], Theorem 1.4 and Proposition 2.3)**.** For every polynomial permutation class C there is a finite set H of peg permutations, each associated with a convex set $\mathcal{V}_\rho$ of filling vectors, such that C is the disjoint union

$$C = \bigsqcup_{\rho \in H} \rho[\mathcal{V}_\rho].$$

In Homberger and Vatter [33, Proposition 2.3] the authors show that every permutation which fills a peg permutation ρ has a unique ρ-partition. Together with Theorem 2.4.2, this leads to the following result.

Theorem 2.4.3. A peg permutation ρ and its convex set $\mathcal{V}_\rho$ of filling vectors is a Struct rule.

Proof. For a peg permutation, ρ, $\mathcal{V}_\rho$ consists of the vectors with $\rho(i) = 1$ when $v(i)$ is dotted and the remaining elements are integers greater than 1. We create ρ' from ρ by replacing an entry x^+ in ρ with the subsequence $(x - 0.2)^\circ(x - 0.1)^\circ x^+$ and entries y^- with the subsequence $(y + 0.2)^\circ(y + 0.1)^\circ y^-$ and taking the standardization of the underlying permutation. The set $\mathrm{Grid}\big(\mathrm{M}_{\rho'}\big)$ is a generalized grid class. Moreover, it is a Struct rule since every permutation which fills ρ has a unique ρ-partition. $\square$

Theorem 2.4.1 follows as a corollary, as we now have a finite set of Struct rules. Given there exists a cover for every polynomial permutation class a natural follow-up question which we do not answer is: for a polynomial permutation class C is there a bound on the size of the Struct rules required in a cover for C?

2.5 Enumerating Struct covers

Most of the contents of this section were implemented in the BSc thesis by Arnarson *et al.* [9] that the author helped advise. With this we have enumerations for every conjectured Struct cover that we have found. In the BSc thesis they discuss a method for getting a recurrence relation, but here we will only explain how to obtain a generating function for a cover.

A Struct cover is a disjoint union of rules. Hence, the generating function is the sum of the generating functions for each rule. Each rule is some matrix whose entries are permutation sets. We assume from here on that the generating function of these permutation sets is known.

For a rule $\mathcal{R}$, we build a graph where the vertices are the cells containing a permutation set. There is an edge between two vertices if the corresponding cells are on the same row or column. If we take the connected components of this graph and translate it back to subrules, then the generating function for $\mathcal{R}$ is the product of the subrules. We call these subrules the *factors* of $\mathcal{R}$. See Figure 2.5 for an example. We have reduced the problem of enumerating covers to enumerating factors. As we assume that our rules have unique griddings, when we reduce it to the factors we must count the number of griddings on the factor, rather than the number of permutations. We adopt this convention throughout the rest of this section.

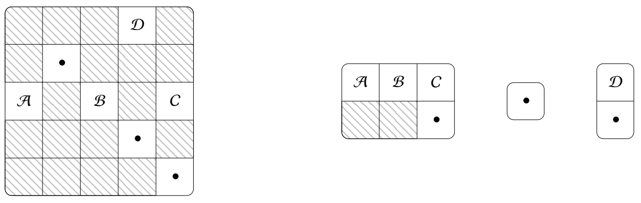


Figure 2.5: An example Struct rule on the left, alongside its factors on the right.

Proposition 2.5.1. For a permutation set $\mathcal{A}$ with generating function $A(x) = \sum_{n\geq 0} a_n x^n$, the rule $\mathcal{R} = \begin{array}{|c|}\hline \mathcal{A} \\ \hline \bullet \\ \hline \end{array}$ is enumerated by the generating function

$$R(x) = x\frac{d}{dx}(xA(x)).$$

Proof. For $n > 0$, the nth coefficient of $x\frac{d}{dx}(xA(x))$ is na_{n-1}. To create a gridding of length $n > 0$ on $\mathcal{R}$ first choose a length $n-1$ permutation from $\mathcal{A}$. The minimum point can be placed at any position between the points in the permutation from $\mathcal{A}$, that is n choices. Hence, there are na_{n-1} griddings on $\mathcal{R}$. □

Proposition 2.5.2. Let $\mathcal{I} = \text{Av}(21)$. The generating function for the rule $\mathcal{R}$ consisting of k $\mathcal{I}$s in a row is $\frac{1}{1-kx}$.

Proof. Consider building a $\mathcal{R}$-gridding by iteratively adding a new maximum. At each iteration, there are k choices where it can go. Hence, there are k^n $\mathcal{R}$-griddings. □

The *Hadamard* product of two generating functions, $A = \sum_{n\geq 0} a_n x^n$ and $B = \sum_{n\geq 0} b_n x^n$, is defined as the pairwise multiplication of terms, $A * B = \sum_{n\geq 0} a_n b_n x^n$. There is a package in Maple which can find the Hadamard product of two holonomic generating functions by Salvy and Zimmermann [51].

Proposition 2.5.3. Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ be permutation sets with generating functions $A_1(x), A_2(x), \ldots, A_k(x)$ respectively and $R(x_1, x_2, \ldots, x_k)$ be the multivariate generating function for the rule $\mathcal{R}$ consisting of the sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ in a single row,

$$\mathcal{R} = \begin{array}{|c|c|c|c|}\hline \mathcal{A}_1 & \mathcal{A}_2 & \cdots & \mathcal{A}_k \\ \hline \end{array},$$

where the coefficient of $x_1^{\ell_1} x_2^{\ell_2} \cdots x_k^{\ell_k}$ is the number of $\mathcal{R}$ griddings with ℓ_i points in the i^{th} column. This generating function is

$$R(x_1, x_2, \ldots, x_k) = \left(\frac{1}{1-(x_1 + x_2 + \cdots + x_k)}\right) *_1 A_1(x_1) *_2 A_2(x_2) *_3 \cdots *_k A_k(x_k)$$

where $*_i$ denotes the hadamard product with respect to the variable x_i. In particular $R(x, x, \ldots, x)$ is the generating function for the number of $\mathcal{R}$-griddings.

Proof. Consider building a gridding on $\mathcal{R}$. First we determine where each value of the gridding goes, i.e. as in 2.5.2 repeatedly add a new maximum. This enumeration is given by $\frac{1}{1-(x_1+x_2+\cdots+x_k)}$. Then in the i^{th} column, the points can be shuffled so long as the values do not change and that it lies within the permutation set $\mathcal{A}_i$, which is equivalent to taking the Hadamard product with $A_i(x_i)$. □

In practice, this is not what we did as our implementation uses SymPy, Meurer *et al.* [49], which, at the time of writing, does not have the Hadamard product implemented. What we do instead is to define a canonical form for rules. In our case, all our permutation sets are points or permutation classes with a known basis. As we are only interested in the enumeration, we replace each basis with its lexicographically smallest symmetry. We then consider the symmetries of the rule and pick the one in which the cells used are lexicographically smallest. We store this canonical form in a database with its generating function which was computed by hand. In total there are less than 100 unique factors found to date, and when a new one is found we add it to our database. The enumeration of our factors is straightforward applications of these (or similar) propositions. For example, the propositions above can be easily extended to handle any factor where there are no subrules of the form $\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}$.

2.6 Results of the Struct algorithm

The enumeration and Wilf-classification are known for all permutation classes with a subset of $\mathcal{S}_3$ for a basis. There exists a Struct cover for all of these classes except Av(123), which we revisit in Chapters 3 and 4.

We will now look at the results of applying the algorithm to bases containing length four patterns. The total number of bases is 2^{24}, but of course, it suffices to look at one basis from each symmetry class. This brings the total down to $2\,097\,152 \approx 2^{21}$ bases. In Section 2.4 we showed that all polynomial permutation classes have a cover. Therefore, we look at the non-polynomial classes, bringing the total down to $157\,736 \approx 2^{17}$ bases.

As expected Struct does better on bases with many patterns. In Table 2.1 we break down the computer output by the size of the basis and the size of the largest dimension of the Struct rules required for the cover. [3] For all of the non-polynomial bases, we searched for a cover with rules of size of at most 7×7. We consider the permutation class failed if Struct does not find a cover within this bound. Of course, there may exist covers using larger rules.

				successes					
$\lvert\Pi\rvert$	non-symmetric	non-polynomial	failures	2×2	3×3	4×4	5×5	6×6	7×7
24	1	0	0	0	0	0	0	0	0
23	7	0	0	0	0	0	0	0	0
22	56	0	0	0	0	0	0	0	0
21	317	0	0	0	0	0	0	0	0
20	1524	0	0	0	0	0	0	0	0
19	5733	1	0	0	1	0	0	0	0
18	17728	9	0	0	8	1	0	0	0
17	44767	58	0	0	32	26	0	0	0
16	94427	285	0	0	75	206	4	0	0
15	166786	1069	0	0	118	901	49	1	0
14	249624	3143	0	0	137	2620	377	9	0
13	316950	7338	0	0	122	5118	2038	60	0
12	343424	13891	0	1	82	6372	7163	273	0
11	316950	21451	1	0	36	4890	15551	970	3
10	249624	27274	12	0	9	2285	21947	2990	31
9	166786	28391	59	0	1	615	19672	7856	188
8	94427	24160	177	6	0	85	9956	13051	885
7	44767	16489	708	0	0	10	2267	10924	2580
6	17728	8935	3249	0	0	2	167	3668	1849
5	5733	3716	2597	0	0	0	7	331	781
4	1524	1187	1160	3	0	0	1	8	15
3	317	279	279	0	0	0	0	0	0
2	56	53	53	0	0	0	0	0	0
1	7	7	7	0	0	0	0	0	0

Table 2.1: Data from running Struct on every non-polynomial basis with length four patterns.

Judging from the data in the table, one would hope to get some successes for bases with three patterns. However, at this point the memory usage becomes infeasible for the computers we have access to, routinely exceeding 32GiB of memory. Therefore, we expect

[3] Most of the computation was done on a cluster owned by Reykjavik University, and the remainder on a cluster owned by the University of Iceland.

there to be some permutation classes with covers consisting of 7×7 rules, although we can not find them at this point.

In Section 6.5, we introduce the notion of an *elementary* permutation class. These are classes which are given by a disjoint union of certain Struct rules with a different block set than used to generate the results in this section. In Table 6.1, we see that a lot of elementary permutation classes that did not find cover.

In Bean, Gudmundsson, and Ulfarsson [16], there are many examples of Struct covers given, including examples which separate Struct and the other automatic methods. The drawback of the Struct algorithm is that it only provides conjectures and it is up to the mathematician to confirm that the cover is correct. The remainder of this thesis will be about removing this step.

Chapter 3

Combinatorial exploration

3.1 Combinatorial classes

The goal of enumerative combinatorics is to enumerate combinatorial objects. One method is to find a *combinatorial specification* as defined in Flajolet and Sedgewick [30, p. 31-33]. We will review some of these definitions briefly. A *combinatorial class* is a set, C, with a size function $C \mapsto \mathbb{N}_0$ such that the preimage for all integers is finite. Any element $c \in C$ is called a *combinatorial object* and the size of c is denoted $|c|$. For a combinatorial class, we let C_n be the set of objects in C of size n.

The set of all permutations $\mathcal{S}$ is a combinatorial class since there are $n!$ permutations of each length n. In Section 2.1, we defined a permutation class as a permutation set closed downwards with respect to the permutation containment order. All permutation sets are combinatorial classes. However, not all are permutation classes. Moreover, subsets of permutation classes are not necessarily permutation classes, but of course are combinatorial classes. In Chapters 4, 5, and 6 we will use the methods of this chapter to enumerate permutation classes.

The goal of this chapter is to enumerate a combinatorial class, that is to determine the sequence $(|C_n|)_{n\geq 0}$. In Section 2.2, we determined this sequence for permutation sets avoiding bivincular patterns and in Section 2.3, we did this for Struct rules. We showed how to construct those combinatorial classes from known combinatorial classes. In this section, we will formalize this method. A common tool we used was generating functions. That is, the ordinary generating function for a combinatorial class C is

$$F_C(x) = \sum_{n\geq 0} |C_n| x^n = \sum_{c \in C} x^{|c|}.$$

In this section, we will only use ordinary generating functions so we will simply refer to these as generating functions.

For two combinatorial classes $\mathcal{A}$ and $\mathcal{B}$, define their *Cartesian product* to be the set of ordered pairs

$$\mathcal{A} \times \mathcal{B} = \{(a, b) \mid a \in \mathcal{A}, b \in \mathcal{B}\},$$

where $|(a, b)| = |a| + |b|$. This is a combinatorial class, and its generating function satisfies

$$F_{\mathcal{A}\times\mathcal{B}}(x) = \sum_{n\geq 0} \sum_{i=0}^{n} |\mathcal{A}_i||\mathcal{B}_{n-i}| x^n = F_{\mathcal{A}}(x) F_{\mathcal{B}}(x).$$

If $\mathcal{A}$ and $\mathcal{B}$ are two disjoint combinatorial classes, then their union is a combinatorial class. We will use the symbol $\sqcup$ to denote the disjoint union. The generating function for

$\mathcal{A} \sqcup \mathcal{B}$ satisfies

$$F_{\mathcal{A}\sqcup\mathcal{B}}(x) = \sum_{n\geq 0} |\mathcal{A}_n|x^n + |\mathcal{B}_n|x^n = F_{\mathcal{A}}(x) + F_{\mathcal{B}}(x).$$

If there is a length preserving bijection $C \mapsto \mathcal{A} \times \mathcal{B}$ or $C \mapsto \mathcal{A} \sqcup \mathcal{B}$ and we know the generating functions for $\mathcal{A}$ and $\mathcal{B}$ then we know the generating function for C. Constructors, such as $\times$ and $\sqcup$, satisfying this property are called admissible constructors. The two discussed here are the only admissible constructors we will consider in this thesis. However, the methods can handle any admissible constructor satisfying the definition outlined by Flajolet and Sedgewick [30, p. 24-30]. When such a length preserving bijection exists we will write $C \cong \mathcal{A}\times\mathcal{B}$ or $C \cong \mathcal{A}\sqcup\mathcal{B}$. The above is naturally extendable to Cartesian products and disjoint unions of any number of combinatorial classes.

A *(combinatorial) rule* is a tuple $(C, \{C_1, C_2, \ldots, C_k\}, \circ)$, where C and all C_i are combinatorial classes and $\circ$ is an admissible constructor, if $C \cong C_1 \circ C_2 \circ \cdots \circ C_k$, that is, there is a length preserving bijection between the C and $C_1 \circ C_2 \circ \cdots \circ C_k$. If the enumeration of a combinatorial class is known then we consider $(C, \emptyset, \circ)$ a combinatorial rule.

A *combinatorial specification* for a tuple $\check{C} = (C_1, C_2, \ldots, C_k)$ of non-empty combinatorial classes is a set of k rules using only combinatorial classes in $\check{C}$ and each C_i appears on the left of a rule exactly once. We say it is a combinatorial specification for C_1 also.

There are two main stages to *combinatorial exploration*. The first is the expansion stage, where *strategies* are applied to the combinatorial classes of interest, resulting in a set of rules which explain how combinatorial classes are related. The second stage is to check if there exists a combinatorial specification for the original combinatorial class of interest within this set of rules. We automated this procedure, and we call our algorithm CombSpecSearcher.

The expansion stage is mostly bookkeeping, and we will discuss how the CombSpecSearcher does this in Section 3.2. In Section 3.3, we will discuss how CombSpecSearcher searches for combinatorial specification in a given set of rules.

3.2 Combinatorial expansion

In the expansion stage of the CombSpecSearcher algorithm there are four types of strategies used. This section will discuss each individually. We will also outline the order in which we apply them, which has been chosen primarily through heuristics. In Chapters 5 and 6 we will discuss such strategies in terms of permutation classes.

The first type of strategy we will discuss we call *expansion* strategies. These take as input a combinatorial class C and return combinatorial rules of the form $(C, S, \circ)$, where S is some set of combinatorial classes. If an expansion strategy returns combinatorial rules with $\circ = \sqcup$ we call it a *batch* strategy and if $\circ = \times$ we call it a *decomposition* strategy.

The CombSpecSearcher expands combinatorial classes in a breadth-first manner. That is, there is a queue which starts with the initial combinatorial class of interest. At each stage, a combinatorial class is removed from the queue and then expanded using all of the expansion strategies. Each combinatorial class in the combinatorial rules returned by this expansion is added to the back of the queue and the process repeats.

An assumption made by the CombSpecSearcher is that there exists a method that determines if a combinatorial class is equal to the empty set. Given that the definition of combinatorial specification requires non-empty combinatorial classes, we remove these from the strategies returned by expansion strategies.

If an expansion strategy returns a combinatorial rule $(C, \{C'\}, \circ)$ then it follows that $C \cong C'$ and we say that the combinatorial classes are *equivalent* as any combinatorial rule containing C will also be a combinatorial rule after exchanging each C with C'. The CombSpecSearcher keeps track of this equivalence using a union-find data structure. The combinatorial rules are considered in terms of the equivalence classes of the combinatorial classes.

If there is a strategy that only returns combinatorial rules that imply equivalence, then we call this an *equivalence* strategy. To try and maximize the size of the equivalence classes of combinatorial classes the CombSpecSearcher always fully expands the equivalence class of every combinatorial class.

When a combinatorial class C is understood, that is there is a method for getting its enumeration, the CombSpecSearcher marks it as verified by giving the combinatorial rule $(C, \emptyset, \circ)$. A strategy for determining this is called a *verification* strategy. Once verified, CombSpecSearcher no longer expands this object[1].

The final type of strategy the CombSpecSearcher uses are called *inferral* strategies. It is an equivalence strategy. However, the goal of an inferral strategy is to attempt to return a canonical form for a combinatorial class. The CombSpecSearcher will only expand further the combinatorial class returned from an inferral strategy and not the original inputted combinatorial class.

The CombSpecSearcher begins with a combinatorial class C and sets of expansion strategies, equivalence strategies, inferral strategies, and verification strategies. It expands in the following manner.

(1) Let Q be a queue containing the combinatorial class C.

(2) Take the first combinatorial class in Q, call it C'

(3) If C' has been expanded previously or is verified then go to Step (2).

(4) Apply all expansion strategies to C', removing all empty combinatorial classes from combinatorial rules

(5) For each combinatorial rule $\mathcal{R}$ first apply inferral strategies to the combinatorial classes in $\mathcal{R}$, second apply verification strategies to each combinatorial class, and finally apply all equivalence strategies to each combinatorial class, ensuring to repeat this step for each equivalent combinatorial class

(6) Add all new combinatorial classes to Q.

(7) If Q is empty, terminate, else go to step (2).

There are a few other search parameters that are allowed by the CombSpecSearcher to customize the search for different combinatorial classes. For example, there might be a reason that some combinatorial classes found should not be expanded in the future[2]. This is added as a boolean and checked in Step (3).

[1]although in some cases there might be a benefit to expanding these

[2]Perhaps a symmetry of a combinatorial class has already been expanded, and the strategies being applied are closed with respect to this symmetry, then no new information would be gained by expanding this symmetric combinatorial class.

3.3 Searching for a combinatorial specification

A combinatorial specification $\check{C} = (C_1, C_2, \ldots, C_k)$ is *iterative* if the combinatorial rules are *upper triangular*, that is they are of the form

$$\begin{aligned}
&(C_1, S_1 \subseteq \{C_2, C_3, \ldots, C_k\}, \circ_1)\\
&(C_2, S_2 \subseteq \{C_3, C_4, \ldots, C_k\}, \circ_2)\\
&\qquad\vdots\\
&(C_{k-1}, S_{k-1} \subseteq \{C_k\}, \circ_{k-1})\\
&(C_k, \emptyset, \circ_k).
\end{aligned}$$

This means that the combinatorial class C_1 can be written as a single equation with known combinatorial classes, that is those whose combinatorial rule is $(C, \emptyset, \circ)$. Otherwise, the combinatorial specification is *recursive*.

Alternatively, define the *dependency graph* of a combinatorial specification, Γ, to be the directed graph with vertices $\{1, 2, \ldots, k\}$ where there is a directed edge $i \mapsto j$ if C_j is in S_i for some combinatorial rule $(C_i, S_i, \circ_i)$. The combinatorial specification is iterative if its dependency graph is acyclic. If the dependency graph of the combinatorial specification contains a cycle, then it is recursive.

After the expansion stage of combinatorial exploration, we have a set of combinatorial rules, $\mathcal{U}$, that we call the *universe*. The CombSpecSearcher searches within this universe to find a combinatorial specification if such exists. We are going to present two algorithms, the first searches for an iterative combinatorial specification, and the second will search for possibly recursive combinatorial specifications.

Algorithm 1 Iterative Combinatorial Specification Searcher

```
1: Input: A set of combinatorial rules $\mathcal{U}$
2: Output: A set of combinatorial rules $\mathcal{U}' \subseteq \mathcal{U}$ such that each combinatorial rule in $\mathcal{U}'$
   is in an iterative combinatorial specification
3:
4: $\mathbf{V} \leftarrow \emptyset$
5: $\mathcal{U}' \leftarrow \emptyset$
6: changed ← True
7: while changed do
8:     changed ← False
9:     for $(C_i, S) \in \mathcal{U} \backslash \mathcal{U}'$ do
10:        if $S \subseteq V$ then
11:            $\mathbf{V} \leftarrow \mathbf{V} \cup \{C_i\}$
12:            $\mathcal{U}' \leftarrow \mathcal{U}' \cup \{(C_i, S)\}$
13:            changed ← True
14:        end if
15:    end for
16: end while
17: return $\mathcal{U}'$
```

Theorem 3.3.1. Let $\mathcal{U}$ be a set of combinatorial rules. A combinatorial rule (C, S) is in an iterative combinatorial specification that is a subset of $\mathcal{U}$ if and only if (C, S) in $\mathcal{U}'$, the set returned by Algorithm 1 with input $\mathcal{U}$.

Proof. Assume (C, S) is in an iterative combinatorial specification with the combinatorial rules $(C_1, S_1 \subseteq \{C_2, C_3, \ldots, C_k\}, \circ_1)$, $(C_2, S_2 \subseteq \{C_3, C_4, \ldots, C_k\}, \circ_2)$, ..., $(C_{k-1}, S_{k-1} \subseteq \{C_k\}, \circ_{k-1})$, and $(C_k, \emptyset, \circ_k)$. On the first pass at least $(C_k, \emptyset, \circ_k)$ will be added to $\mathcal{U}'$ and C_k added to $\mathbf{V}$. On the i^{th} pass, C_k, ..., C_{k-i-1} will be on the left hand side of some combinatorial rule. Therefore at this stage $S_{k-i} \subseteq \mathbf{V}$, so if not already in $\mathcal{U}'$ $(C_{k-i}, S_{k-1}, \circ_{k-i})$ it will be added. Hence by induction all of the combinatorial rules will be in $\mathcal{U}'$.

When a rule is added to $\mathcal{U}'$, it is in some iterative combinatorial specification, so it follows that all combinatorial rules in $\mathcal{U}'$ are in a combinatorial specification. □

Algorithm 2 Combinatorial Specification Searcher

```
 1: Input: A set of combinatorial rules $\mathcal{U}$
 2: Output: A set of combinatorial rules $\mathcal{U}' \subseteq \mathcal{U}$ such that each combinatorial rule in $\mathcal{U}'$
    is in a combinatorial specification
 3:
 4: changed ← True
 5: while changed do
 6:     changed ← False
 7:     for $(C_i, S) \in \mathcal{U}$ do
 8:         if any $C_j \in S$ not in some $(C_j, S') \in \mathcal{U}$ then
 9:             $\mathcal{U} \leftarrow \mathcal{U} \setminus \{(C_i, S)\}$
10:             changed ← True
11:         end if
12:     end for
13: end while
14: return $\mathcal{U}$
```

Theorem 3.3.2. Let $\mathcal{U}$ be a set of combinatorial rules. A combinatorial rule $(C, S, \circ)$ is in a combinatorial specification that is a subset of $\mathcal{U}$ if and only if $(C, S, \circ)$ is in $\mathcal{U}'$, the set returned by Algorithm 2 with input $\mathcal{U}$.

Proof. Assume (C, S) is in a combinatorial specification with the combinatorial rules $(C_1, S_1, \circ_1)$, $(C_2, S_2, \circ_2)$, ..., $(C_k, S_k, \circ_k)$. Each S_i is a subset of $\{C_1, C_2, \ldots, C_k\}$. Therefore, at each pass all of the combinatorial rules will not be removed from $\mathcal{U}$. Hence, this combinatorial specification will still be a subset of $\mathcal{U}$ at the end of the algorithm.

For a given combinatorial rule (C, S) in $\mathcal{U}$ at the end of the algorithm it follows, every combinatorial class C' in S is in some combinatorial rule. Pick one, and repeat unless the class C' is already on the left-hand side of some rule chosen. This procedure is guaranteed to terminate since the set is finite. □

These two algorithms allow searching for combinatorial specifications. During the combinatorial expansion, as in Section 3.2, we periodically check the universe found to find a combinatorial specification using these algorithms. It is then easy to tell if any combinatorial class is in a combinatorial specification, so the CombSpecSearcher treats those that are as verified and expands them no further unless otherwise instructed.

Chapter 4

Gridded permutations

4.1 Figures and tilings

In this thesis, we have been thinking of permutations with a geometric mindset. In Section 2.3, we introduced Struct rules, grids used to represent permutation sets. To do this, we defined a gridding on the rules and then only allowed rules in which permutations had a unique gridding. The Struct rules we used for permutation classes use subclasses as the permutation sets allowed in the cells. In this section, we explore allowing restrictions across cells, in terms of some gridding. Moreover, we are going to drop the condition for unique griddings, and rather than consider permutations with a gridding we will consider the set of all griddings.

In order to do this, we borrow some definitions from Section 2 of Albert *et al.* [7] altering them slightly to suit our thinking. The *grid position* of a point (x, y) in $\mathbb{R}^2$ is defined as the integer point (a, b) such that $(x, y) \in [a, a + 1) \times [b, b + 1)$.

We say that a *figure* $\mathcal{F} \subseteq \mathbb{R}^2$ is *grid involved* in the figure $\mathcal{G}$, denoted $\mathcal{F} \leq \mathcal{G}$ if there are subsets $A, B \subseteq \mathbb{R}$ and increasing injections $\phi_x : A \mapsto \mathbb{R}$ and $\phi_y : B \mapsto \mathbb{R}$ such that

$$\mathcal{F} \subseteq A \times B \text{ and } \phi(\mathcal{F}) = \{(\phi_x(a), \phi_y(b)) : (a, b) \in \mathcal{F}\} \subseteq \mathcal{G}$$

and ϕ preserves the grid position of a point (x, y) in $\mathbb{R}^2$. Ignoring the grid position recovers the definition of involvement as in Albert *et al.* [7].

As in Albert *et al.* [7] this relation forms a preorder on the collection of all figures. If $\mathcal{F} \leq \mathcal{G}$ and $\mathcal{G} \leq \mathcal{F}$ we say they are *grid equivalent*. (This means two figures are equivalent if and only if one can be transformed to the other by stretching and shrinking the axes without ever crossing the integer lattice).

We say a figure is *independent* if no two points lie on the same horizontal or vertical line. The set of all *gridded permutations*, $\mathcal{G}$, is then defined as the set of all equivalence classes of finite independent figures with respect to grid equivalence. We define containment of gridded permutations with respect to the involvement relation, and we refer to σ as a gridded pattern when $\sigma \leq \pi$ for some gridded permutation π. Define the *length* of a gridded permutation as the number of points. The standardization of the set of sorted points is called the *underlying permutation*.

For convenience to represent an equivalence class we write the underlying permutation where we put the grid position of each point in its exponent, for example the figure $\left\{\left(\frac{1}{2}, \frac{1}{2}\right), \left(\frac{3}{2}, \frac{3}{2}\right)\right\}$ is in the equivalence class $1^{(0,0)}2^{(1,1)}$. A larger example is given in Figure 4.1 where we have drawn the figure in Equation (4.1).

$$\mathcal{F} = \left\{\left(\frac{1}{4}, \frac{3}{4}\right), \left(\frac{3}{4}, \frac{15}{4}\right), \left(\frac{5}{4}, \frac{7}{4}\right), \left(\frac{7}{4}, \frac{5}{4}\right), \left(\frac{9}{4}, \frac{13}{4}\right), \left(\frac{11}{4}, \frac{11}{4}\right), \left(\frac{13}{4}, \frac{17}{4}\right), \left(\frac{15}{4}, \frac{1}{4}\right), \left(\frac{15}{4}, \frac{9}{4}\right)\right\}. \tag{4.1}$$

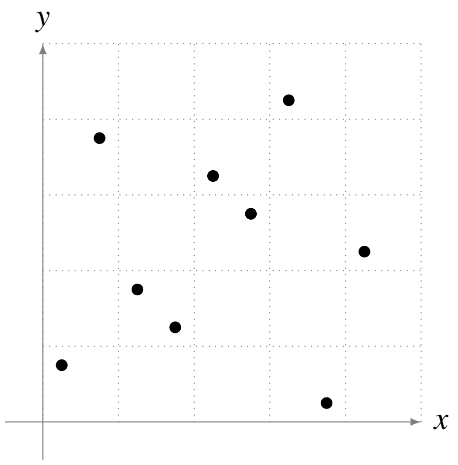


Figure 4.1: A drawing of the figure $\mathcal{F}$ from Equation (4.1) that is in the equivalence class represented by $2^{(0,0)}8^{(0,3)}4^{(1,1)}3^{(1,1)}7^{(2,3)}6^{(2,2)}9^{(3,4)}1^{(3,0)}5^{(4,2)}$.

The set of gridded permutations $\mathcal{G}$ is not a combinatorial class, as there are not finitely many gridded permutations of each length. Therefore, we define $\mathcal{G}^{(n,m)}$ to be the set of gridded permutations with grid positions in $\{0,\ldots,n-1\}\times\{0,\ldots,m-1\}$. This set is a combinatorial class as there are $k!\binom{n+k-1}{n-1}\binom{m+k-1}{m-1}$ length k gridded permutations with grid positions in $\{0,\ldots,n-1\}\times\{0,\ldots,m-1\}$. The binomials in this formula count the number of ways to draw $n-1$ horizontal and $m-1$ vertical lines into the $k+1$ positions between the points of a length k permutation.

A gridded permutation *avoids* a set of gridded patterns $\mathcal{O}$ if it avoids every gridded permutation in $\mathcal{O}$. Let

$$\mathrm{Av}^{(n,m)}(\mathcal{O}) = \{\pi \in \mathcal{G}^{(n,m)} \mid \pi \text{ avoids } \mathcal{O}\}$$

be the *avoiders* of $\mathcal{O}$, and $\mathrm{Av}_k^{(n,m)}(\mathcal{O})$ be the length k gridded permutations in this set. We say a gridded permutation *contains* a set of gridded patterns $\mathcal{R}$ if it does not avoid $\mathcal{R}$. That is it contains at least one gridded permutation in $\mathcal{R}$. The *containers* of $\mathcal{R}$ is the set

$$\mathrm{Co}^{(n,m)}(\mathcal{R}) = \{\pi \in \mathcal{G}^{(n,m)} \mid \pi \text{ contains } \mathcal{R}\}$$

and $\mathrm{Co}_k^{(n,m)}(\mathcal{R})$ is the set of length k gridded permutations in this set.

Lemma 4.1.1. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two finite sets of gridded patterns. Then there exists a finite set $\mathcal{R}$ such that

$$\mathrm{Co}^{(n,m)}(\mathcal{R}) = \mathrm{Co}^{(n,m)}(\mathcal{R}_1) \cap \mathrm{Co}^{(n,m)}(\mathcal{R}_2).$$

For the proof of this lemma we define the *merge* of two gridded permutations σ and τ to be the set $\texttt{merge}(\sigma,\tau)$ of gridded permutations which contain both σ and τ but contain no smaller such gridded permutation, i.e.,

$$\texttt{merge}(\sigma,\tau) = \{\pi \in \mathrm{Co}(\sigma)\cap\mathrm{Co}(\tau) \mid \forall\delta \in \mathcal{G} \text{ such that } \delta \prec \pi, \delta \notin \mathrm{Co}(\sigma)\cap\mathrm{Co}(\tau)\}\,.$$

The proof shows that $\texttt{merge}(\sigma,\tau)$ is finite, and moreover one method to generate this set is to generate all gridded permutations to the bound $|\sigma|+|\tau|$ and filter.

Proof. Let π be a gridded permutation in $\mathrm{Co}(\sigma)\cap\mathrm{Co}(\tau)$, so π contains an occurrence of both σ and τ. Select an occurrence of each and by removing the remaining points create a smaller gridded permutation in $\mathrm{Co}(\sigma) \cap \mathrm{Co}(\tau)$ of length at most $|\sigma| + |\tau|$. Hence, the length of the elements in $\texttt{merge}(\sigma, \tau)$ have length at most $|\sigma| + |\tau|$. In particular, the set $\texttt{merge}(\sigma, \tau)$ is finite.

Let $\mathcal{R} = \bigcup_{(\sigma,\tau)\in\mathcal{R}_1\times\mathcal{R}_2} \texttt{merge}(\sigma, \tau)$. Then it follows that

$$\mathrm{Co}^{(n,m)}(\mathcal{R}) = \mathrm{Co}^{(n,m)}(\mathcal{R}_1) \cap \mathrm{Co}^{(n,m)}(\mathcal{R}_2)$$

since π contains $\mathcal{R}_1$ and $\mathcal{R}_2$ if π contains some σ in $\mathcal{R}_1$ and τ in $\mathcal{R}_2$, which implies π contains some gridded permtutation in the $\texttt{merge}(\sigma, \tau)$. Conversely, if π contains a gridded permutation in $\mathcal{R}$, then this is in some set $\texttt{merge}(\sigma, \tau)$ for some σ in $\mathcal{R}_1$ and τ in $\mathcal{R}_2$. This implies that π contains σ and τ and in particular contains the sets $\mathcal{R}_1$ and $\mathcal{R}_2$. □

We define a *tiling* to be a triple $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k\})$ where $\mathcal{O}$ is a set of gridded permutations that we call *obstructions*, and $\mathcal{R}$ a set of sets of gridded permutations that it must contain that we call *requirements*. We define

$$\mathrm{Grid}(\mathcal{T}) = \mathrm{Av}^{(n,m)}(\mathcal{O}) \cap \mathrm{Co}^{(n,m)}(\mathcal{R}_1) \cap \mathrm{Co}^{(n,m)}(\mathcal{R}_2) \cap \cdots \cap \mathrm{Co}^{(n,m)}(\mathcal{R}_k).$$

Denote $\mathrm{Grid_n}(\mathcal{T})$ to be the length n gridded permutations in $\mathrm{Grid}(\mathcal{T})$. When discussing tilings referring to an *obstruction* will mean a gridded permutation we must avoid, and a *requirement* will mean a set of gridded permutations that we must contain.

Lemma 4.1.1 shows that the containment of any number of sets of gridded permutation can be reduced to a single set. Therefore for every tiling $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k\})$, there exists a single set of gridded permutations $\mathcal{R}'$ such that

$$\mathrm{Grid}(\mathcal{T}) = \mathrm{Av}^{(n,m)}(\mathcal{O}) \cap \mathrm{Co}^{(n,m)}(\mathcal{R}').$$

However, it is often more convenient to think about requiring many smaller requirements and this is why we have chosen to use the more fine-grained definition. In some cases, this simpler definition can prove easier to work with.

For example, consider the tiling

$$\mathcal{T} = \left((1, 1), \left\{1^{(0,0)}2^{(0,0)}, 2^{(0,0)}1^{(0,0)}\right\}, \left\{\left\{1^{(0,0)}\right\}\right\}\right). \tag{4.2}$$

In Figure 4.2, there is a pictorial representation of $\mathcal{T}$ where obstructions have been drawn in red and the requirement drawn in green. With this tiling it is clear that

$$\mathrm{Grid}(\mathcal{T}) = \left\{1^{(0,0)}\right\}.$$

We therefore call this the *point tiling*.

We say a gridded permutation is *local* if the grid positions of all the points are the same. We write π^c to represent the local gridded permutation with underlying pattern π with grid positions being c.

Proposition 4.1.2. Let $C = \mathrm{Av}(\Pi)$ be a permutation class and

$$\mathcal{T} = \left((1, 1), \{\sigma^{(0,0)} \mid \sigma \in \Pi\}, \emptyset\right),$$

then $\mathrm{Grid}(\mathcal{T})$ is in bijection with C.

Proof. The mapping $\phi : C \mapsto \mathrm{Grid}(\mathcal{T})$ given by $\phi(\pi) = \pi^{(0,0)}$ is a bijection. □

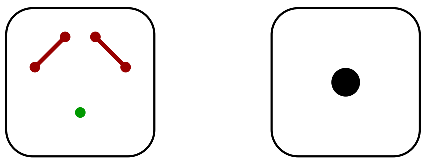


Figure 4.2: The left hand picture is a representation of the tiling $\mathcal{T}$ in Equation (4.2), called the point tiling. As the set Grid($\mathcal{T}$) contains only the gridded permutation $1^{(0,0)}$ we draw it as a point as in the right hand side tiling.

The goal is to use the methods of Chapter 3 to find a combinatorial specification for tilings. We will translate our geometric proof ideas for permutation classes into the language of gridded permutations. We will discuss this in greater detail in Chapters 5 and 6, and then by using this proposition we will be able to find combinatorial specifications for and enumerate permutation classes. We will first develop a theory for gridded permutations and tilings. These will be used implicitly in later sections and appear in the CombSpecSearcher algorithm as inferral strategies.

4.2 Minimal obstructions and requirements

Let σ and τ be two gridded permutations such that $\sigma \preceq \tau$. If a gridded permutation π avoids σ then it also avoids τ. If π contains τ then it clearly contains σ. Imagine σ and τ are in some set $\mathcal{O} \subseteq \mathcal{G}$. Then

$$\mathrm{Av}^{(n,m)}(\mathcal{O}) = \mathrm{Av}^{(n,m)}(\mathcal{O}\backslash\{\tau\}).$$

since avoidance of σ implies the avoidance τ. Similarly, if σ and τ are in some set $\mathcal{R} \subseteq \mathcal{G}$, then

$$\mathrm{Co}^{(n,m)}(\mathcal{R}) = \mathrm{Co}^{(n,m)}(\mathcal{R}\backslash\{\tau\}),$$

since the containment of τ implies the containment of σ. Let $\mathcal{T} = ((n,m), \mathcal{O}, \{\mathcal{R}_1, \dots, \mathcal{R}_k\})$ be a tiling such that $\sigma \in \mathcal{O}$ and $\tau \in \mathcal{R}_i$ for some i. Since avoidance of σ implies the avoidance of τ we can freely remove τ from $\mathcal{R}_i$ as we can not contain it[1]. Moreover, we can replace $\mathcal{R}_i$ with the set of gridded permutations in $\mathcal{R}_i$ that avoid $\mathcal{O}$. Let

$$\mathcal{R}'_i = \mathcal{R}_i \cap \mathrm{Av}(\mathcal{O}) \text{ and } \mathcal{T}' = \left((n,m), \mathcal{O}, \{\mathcal{R}'_1, \dots, \mathcal{R}'_k\}\right),$$

then it follows that Grid($\mathcal{T}$) = Grid($\mathcal{T}'$).

For a gridded permutation, let $\Delta(\pi)$ be the set of all gridded patterns contained in π,

$$\Delta(\pi) = \{\sigma \in \mathcal{G} : \sigma \preceq \pi\},$$

and define the *intersection* of two gridded permutations π_1 and π_2 to be the set of gridded permutations that both contain,

$$\pi_1 \cap \pi_2 = \{\sigma \mid \sigma \in \Delta(\pi_1) \cap \Delta(\pi_2)\}.$$

[1]If on removing τ from $\mathcal{R}_i$ this set becomes empty, then there are no gridded permutations that can satisfy the requirements, since $\mathrm{Co}(\emptyset) = \emptyset$.

For example, the intersection of the gridded permutations $\pi_1 = 1^{(0,0)}3^{(0,0)}2^{(1,0)}4^{(1,1)}$ and $\pi_2 = 1^{(0,0)}2^{(0,0)}3^{(1,0)}4^{(1,1)}$ is

$$\begin{aligned}\pi_1 \cap \pi_2 = \{&\epsilon, 1^{(0,0)}, 1^{(1,0)}, 1^{(1,1)}, 1^{(0,0)}2^{(0,0)}, 1^{(0,0)}2^{(1,0)}, \\ &1^{(0,0)}2^{(1,1)}, 1^{(0,0)}2^{(0,0)}3^{(1,0)}, 1^{(0,0)}2^{(1,0)}3^{(1,1)}\}.\end{aligned} \tag{4.3}$$

This can be seen pictorially in Figure 4.3.

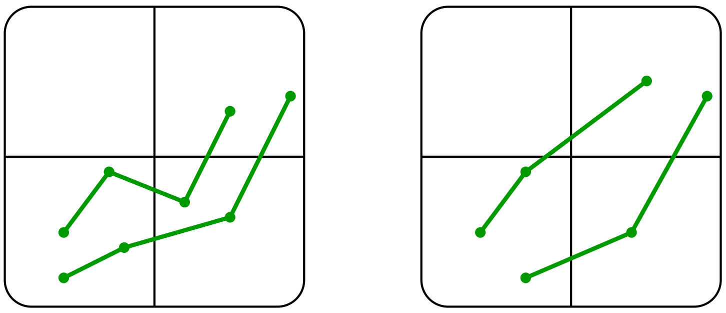


Figure 4.3: The left hand tiling $\mathcal{T}_1$ has the gridded permutations $\pi_1 = 1^{(0,0)}3^{(0,0)}2^{(1,0)}4^{(1,1)}$ and $\pi_2 = 1^{(0,0)}2^{(0,0)}3^{(1,0)}4^{(1,1)}$ as the requirement $\mathcal{R}_1 = \{\pi_1, \pi_2\}$. The right hand tiling $\mathcal{T}_2$ has the maximal elements of intersection of π_1 and π_2 as two size one requirements. By Proposition 4.2.1 we know $\text{Grid}(\mathcal{T}_1) \subseteq \text{Grid}(\mathcal{T}_2)$.

Proposition 4.2.1. For a set of gridded permutations $\mathcal{R}$, the containers of $\mathcal{R}$ is a subset of the containers of each gridded permutation in the intersection of the gridded permutations in $\mathcal{R}$, i.e.

$$\text{Co}(\mathcal{R}) \subseteq \text{Co}(\tau) \text{ for all } \tau \text{ in } \bigcap_{\sigma \in \mathcal{R}} \sigma.$$

Proof. Let π in $\text{Co}(\mathcal{R})$. There is some τ in $\mathcal{R}$ such that $\tau \leq \pi$, and π contains every element of $\Delta(\tau)$. In particular, π contains every element in the intersection $\bigcap_{\sigma \in \mathcal{R}} \sigma$, since it is a subset of $\Delta(\tau)$. □

A gridded permutation σ is a *factor* of a gridded permutation π if π has an occurrence of σ and the gridded positions of points in the complement of this occurrence do not share a row or column with any of the gridded positions of the points in the occurrence. The *factorization* of π is the set $\mathbb{F}(\pi)$ of minimal factors it contains. For example, the factorization of the gridded permutation $\pi = 1^{(0,0)}2^{(0,0)}5^{(1,2)}6^{(1,2)}3^{(2,1)}4^{(3,2)}7^{(3,3)}$ is

$$\mathbb{F}(\pi) = \left\{1^{(0,0)}2^{(0,0)}, 2^{(1,2)}3^{(1,2)}1^{(3,2)}4^{(3,3)}, 1^{(3,1)}\right\}.$$

This can be seen pictorially in Figure 4.4.

Lemma 4.2.2. For a gridded permutation σ, $\text{Co}(\sigma) = \bigcap_{\tau \in \mathbb{F}(\sigma)} \text{Co}(\tau)$.

Proof. Assume π contains σ. Then for $\tau \leq \sigma$ it follows $\tau \leq \pi$, hence $\text{Co}(\sigma) \subseteq \bigcap_{\tau \in \mathbb{F}(\sigma)} \text{Co}(\tau)$. Conversely, assume π avoids σ then, as the factorization of π covers π, there is some factor τ that π avoids, so $\pi \notin \text{Co}(\tau)$ and not in the intersection. □

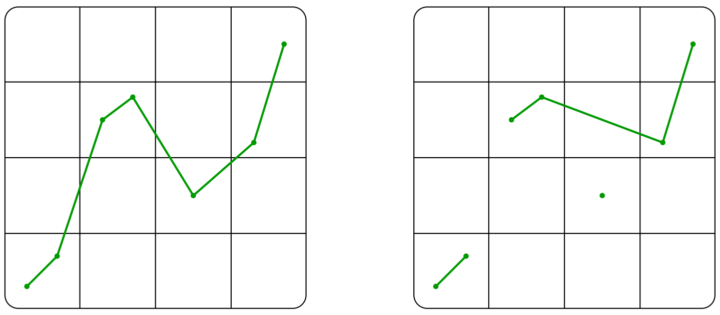


Figure 4.4: The left tiling $\mathcal{T}_1$ has the requirement $\left\{\pi = 1^{(0,0)}2^{(0,0)}5^{(1,2)}6^{(1,2)}3^{(2,1)}4^{(3,2)}7^{(3,3)}\right\}$, and the right tiling $\mathcal{T}_2$ has a requirement for each factor in $\mathbb{F}(\pi)$. By Lemma 4.2.2, we know $\mathrm{Grid}(\mathcal{T}_1) = \mathrm{Grid}(\mathcal{T}_2)$.

The containment of a gridded permutation is equivalent to the containment of the factors. We are now going to discuss removing a factor from a gridded permutation. Let σ and π be two gridded permutations such that σ is a factor of π. Define $\pi\backslash\sigma$ to be the gridded permutation obtained by removing the single occurrence of σ. For example,

$$1^{(0,0)}2^{(0,0)}5^{(1,2)}6^{(1,2)}4^{(2,2)}7^{(2,2)}3^{(3,1)}\backslash 2^{(1,2)}3^{(1,2)}1^{(2,2)}4^{(2,2)} = 1^{(0,0)}2^{(0,0)}3^{(3,1)}.$$

If σ is not a factor define $\pi\backslash\sigma = \pi$. For a tiling $\mathcal{T}$ with $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Co}(\sigma)$ it is possible to remove σ from the obstructions and requirements if it appears as a factor.

Proposition 4.2.3. Let σ be a gridded permutation and $\mathcal{O}, \mathcal{R} \subseteq \mathcal{G}$, then

$$\mathrm{Av}(\mathcal{O}) \cap \mathrm{Co}(\sigma) = \mathrm{Av}(\{\tau\backslash\sigma \mid \tau \in \mathcal{O}\}) \cap \mathrm{Co}(\sigma) \tag{4.4}$$

and if all τ in $\mathcal{R}$ contain σ as a factor

$$\mathrm{Co}(\mathcal{R}) \cap \mathrm{Co}(\sigma) = \mathrm{Co}(\{\tau\backslash\sigma \mid \tau \in \mathcal{R}\}) \cap \mathrm{Co}(\sigma). \tag{4.5}$$

Proof. Let π be a gridded permutation that contains σ.

1. If τ in $\mathcal{O}$ does not contain σ as a factor then $\tau\backslash\sigma = \tau$ and the avoidance condition does not change. If τ does contain σ as a factor, then an occurrence of $\tau\backslash\sigma$ in π together with an occurrence of σ will form an occurrence of τ in π, hence τ avoids $\tau\backslash\sigma$. Conversely, avoiding $\tau\backslash\sigma$ implies avoiding τ.

2. Assume all τ in $\mathcal{R}$ contain σ as a factor, so $\tau\backslash\sigma$ is a factor of τ for each τ. If π contains τ then π contains $\tau\backslash\sigma$. In order to satisfy that at least one τ is contained π must contain at least one $\tau\backslash\sigma$. Conversely, an occurrence of $\tau\backslash\sigma$ together with an occurrence of σ forms an occurrence of τ.

□

Consider the tiling $\mathcal{T} = ((3,1), \{1^{(0,0)}2^{(1,0)}, 1^{(1,0)}2^{(2,0)}\}, \{\{1^{(1,0)}\}\})$, shown in Figure 4.5. For a gridded permutation in $\mathrm{Grid}(\mathcal{T})$ all of the points in $(0,0)$ must be below the points in $(1,0)$. Similarly, the points in $(1,0)$ must be below the points in $(2,0)$. Since $(1,0)$ contains a point, it follows that all of the points in $(0,0)$ are below the points in $(2,0)$. In particular, the gridded permutation avoids $1^{(0,0)}2^{(2,0)}$. Looking at the set of obstructions and requirements, it is not clear, without this argument, why the gridded permutations in $\mathrm{Grid}(\mathcal{T})$ avoid $1^{(0,0)}2^{(2,0)}$, in particular notice that the sets $\{1^{(0,0)}2^{(1,0)}, 1^{(1,0)}2^{(2,0)}\}$ and $\{1^{(0,0)}2^{(1,0)}, 1^{(1,0)}2^{(2,0)}, 1^{(0,0)}2^{(2,0)}\}$ are both minimal with respect to set containment. It is, however, possible to check if any gridded permutation is avoided by a tiling.

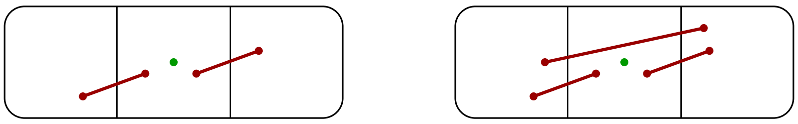


Figure 4.5: Two tilings that have the same set of gridded permutations but have two different minimal sets of obstructions with respect to set containment.

Theorem 4.2.4. For a gridded permutation π and tiling $\mathcal{T}$ it is decidable if $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Av}(\pi)$.

Proof. If $\mathcal{R} = \emptyset$ set $\mathcal{R} = \{\{\epsilon\}\}$. By Lemma 4.1.1 a tiling can be represented as $\mathcal{T} = ((n,m), \mathcal{O}, \mathcal{R})$ where $\mathcal{O}$ and $\mathcal{R}$ are two sets of gridded permutations. In the proof of Lemma 4.1.1 we introduced the merge of two gridded permutations. We will show that $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Av}(\pi)$ if and only if for all σ in $\mathcal{R}$ all gridded permutations in $\mathtt{merge}(\pi, \sigma)$ contain some τ in $\mathcal{O}$. Decidability follows since $\mathtt{merge}(\pi, \sigma)$ is a finite set.

Assume $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Av}(\pi)$. For each σ in $\mathcal{R}$, a permutation π' in $\mathtt{merge}(\pi, \sigma)$ contains π and contains $\mathcal{R}$. As $\mathrm{Grid}(\mathcal{T}) = \mathrm{Av}^{(n,m)}(\mathcal{O}) \cap \mathrm{Co}^{(n,m)}(\mathcal{R})$ it follows that π' is not in $\mathrm{Av}^{(n,m)}(\mathcal{O})$, that is it contains some τ in $\mathcal{O}$.

Conversely, assume $\mathrm{Grid}(\mathcal{T}) \nsubseteq \mathrm{Av}(\pi)$ and let $\delta \in \mathrm{Grid}(\mathcal{T})$ be such that δ contains π. There is some $\sigma \in \mathcal{R}$ such that $\sigma \leq \delta$. Then the permutation $\delta' \leq \delta$ created by taking the union of the occurrences of σ and π in δ' is an element of $\mathtt{merge}(\pi, \sigma)$. Since $\delta \in \mathrm{Av}^{(n,m)}(\mathcal{O})$ so is δ'. Hence, δ' does not contain any τ in $\mathcal{O}$. □

Corollary 4.2.5. For a tiling $\mathcal{T}$ it is decidable if $\mathrm{Grid}(\mathcal{T})$ contains no gridded permutations.

Proof. Check if $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Av}^{(n,m)}(\epsilon)$. □

We have shown already that there can be two sets $\mathcal{O}$ and $\mathcal{O}'$ that are minimal with respect to set containment such that $\mathcal{T} = ((n,m), \mathcal{O}, \mathcal{R})$ and $\mathcal{T}' = ((n,m), \mathcal{O}', \mathcal{R})$. In some sense, Lemma 4.1.1 gives a canonical form for the set of requirements. Is there a useful canonical form for the set of obstructions? Moreover, if it does exist, how can it be computed?

4.3 Counting tilings

Let us briefly step away from gridded permutations. Consider the set of permutations $\mathrm{Av}_n(\Pi) \cap \mathrm{Co}_n(\sigma)$ for some set Π of permutations and some permutation σ. To count the size of this set, we can use the principle of inclusion and exclusion. That is

$$|\mathrm{Av}_n(\Pi) \cap \mathrm{Co}_n(\sigma)| = |\mathrm{Av}_n(\Pi)| - |\mathrm{Av}_n(\Pi \cup \{\sigma\})|, \tag{4.6}$$

which, of course, generalizes to containing many patterns, i.e.

$$|\mathrm{Av}_n(\Pi) \cap \mathrm{Co}_n(\sigma_1) \cap \cdots \cap \mathrm{Co}_n(\sigma_k)| = \sum_{S \subseteq \{\sigma_1, \ldots, \sigma_k\}} (-1)^{|S|} |\mathrm{Av}_n(\Pi \cup S)|.$$

Therefore, if we assume we can enumerate $|\mathrm{Av}_n(\Pi)|$ and all of its subclasses, then we have the enumeration of this set where we force the containment of certain patterns.

In Section 2.3 we covered a method to enumerate Struct rules. The method generalized to counting the number of griddings on any $n \times m$ grid where the enumeration of permutation sets in each cell is assumed. The first step was to reduce the Struct rules into factors. We will do the same for tilings. This method will also generalize to a combinatorial rule with respect to the Cartesian product.

The *active cells* of a tiling are all those who do not contain a length 1 obstruction. We will create the graph whose vertices correspond to these active cells and say that two cells are connected if there is an obstruction or requirement which uses both cells, or the two cells are on the same row and column. We will then say that the *factors* of the tiling are the tilings that are created by using the obstructions and requirements in each connected component (the cells not mentioned in any gridded permutation are assumed to contain a length 1 obstruction). An example of factors is given in Figure 4.6.

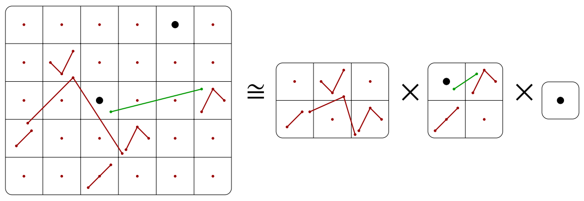


Figure 4.6: A tiling and its factors.

It is clear that if $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ are the factors of a tiling $\mathcal{T}$ then

$$(\mathrm{Grid}(\mathcal{T}), \{\mathrm{Grid}(\mathcal{T}_1), \mathrm{Grid}(\mathcal{T}_2), \ldots, \mathrm{Grid}(\mathcal{T}_k)\}, \times)$$

is a combinatorial rule. We define Factors to be the strategy that takes as input a tiling and returns this combinatorial rule. For reasons that will become apparent in later sections we also say the strategy returns all combinations of factors.

Let $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R})$ be a tiling, where all obstructions in $\mathcal{O}$ are local. If $\mathcal{R} = \emptyset$ then we can apply the methods of Section 2.3. We can enumerate sets with containment, and so the method generalizes to allowing $\mathcal{R}$ to contain length 1 requirements that are local. We call tilings that satisfy this *subset verified* since we assume we can get the enumeration for it. In Figure 4.7, all of the tilings in this Struct cover are subset verified.

We say a gridded permutation is *locally factorable* if all of the factors in its factorization are local. From Lemma 4.2.2 the containment of a locally factorable gridded permutation is equivalent to containing all of the factors.

Consider a locally-factorable gridded permutation. It has the form

$$\pi = \left(\pi_1^{c_1} \cdots \pi_{i_1}^{c_1}\right) \left(\pi_{i_1+1}^{c_2} \cdots \pi_{i_2}^{c_2}\right) \cdots \left(\pi_{i_{k-1}+1}^{c_k} \cdots \pi_{i_k}^{c_k}\right)$$

where each of the c_i are on their own row and column. Then if we consider the gridded permutations avoiding π, they satisfy the formula

$$\mathrm{Av}(\pi) = \left(\mathrm{Av}\left(\pi_{i_1+1}^{c_2} \cdots \pi_{i_2}^{c_2} \cdots \pi_{i_{k-1}+1}^{c_k} \cdots \pi_{i_k}^{c_k}\right) \cap \mathrm{Co}\left(\pi_1^{c_1} \cdots \pi_{i_1}^{c_1}\right)\right) \sqcup \mathrm{Av}\left(\pi_1^{c_1} \cdots \pi_{i_1}^{c_1}\right),$$

in particular if we iterate this procedure we end with a formula for $\mathrm{Av}(\pi)$ in terms of avoidance and containment of local gridded permutations. We then have a method to enumerate

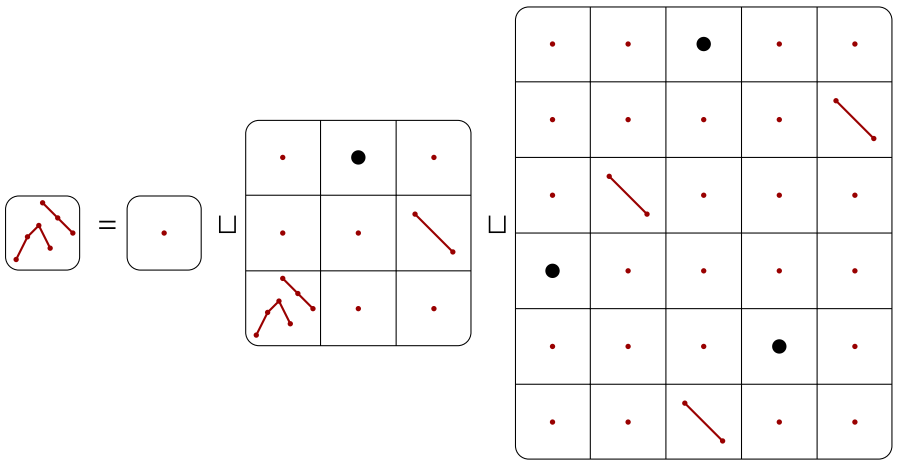
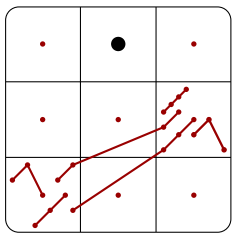


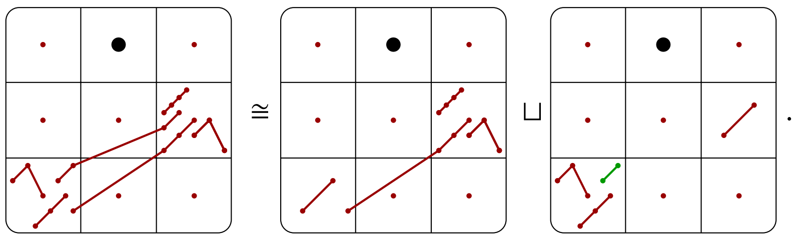
Figure 4.7: The Struct cover from Proposition 2.3.2 in terms of tilings.

tilings where all of the obstructions are locally factorable, assuming the enumeration of the appropriate permutation classes are known. We call a tiling $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R})$ *locally factorable* if all obstructions in $\mathcal{O}$ are locally factorable and all of the requirements in $\mathcal{R}$ are sets of size 1 containing a local gridded permutation. We define the verification strategy LocallyFactorable that checks if a tiling is locally factorable as we have a method to enumerate such tilings.

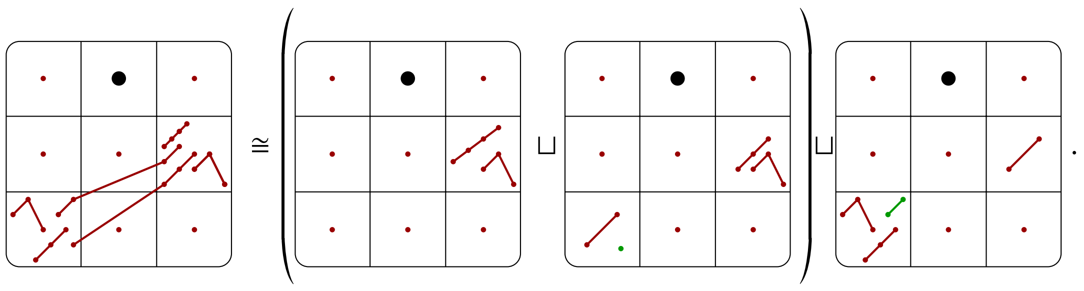
For example, consider the locally factorable tiling



which represents the set of non-empty permutations in Av(231, 1234) as can be seen from the methods we will introduce in Chapter 6. The gridded permutation $1^{(0,0)}2^{(0,0)}3^{(2,1)}4^{(2,1)}$ is locally factorable. In particular it contains the local factor $1^{(0,0)}2^{(0,0)}$, and by Equation (4.3), we get that



Applying Equation (4.3) to the first tiling on the right hand side gives



(4.7)

All of the gridded permutations in the tilings in Equation (4.7) are local. Therfore, it can be easily enumerated using Equation (4.6) and the methods in Section 2.5.

Chapter 5

Revisiting the insertion encoding

5.1 Insertion encoding

In this section, we will review the *insertion encoding* introduced by Albert, Linton, and Ruškuc [6]. The underlying idea is that every length n permutation can be created by taking a length $n-1$ permutation and inserting a new maximum. This is also true for the permutations in a permutation class however not all positions are allowed. Tracing how to build a permutation in this way starting from the empty permutation is called the *evolution* of a permutation. In the insertion encoding, as we map the evolution, we keep track of where future points will be inserted. For example, Figure 5.1 has the evolution of the permutation 325146 where the $\diamond$ is viewed as the promise of a future point. A $\diamond$ is called a *slot* and a string such as $32\diamond1\diamond$ is called a *configuration*. We can not continue to evolve from a configuration with no slots. The idea is to model the evolution using a language.

$$\begin{array}{c}
\diamond \\
\diamond\ 1\ \diamond \\
\diamond\ 2 \diamond\ 1\ \diamond \\
3\ 2 \diamond 1\ \diamond \\
3\ 2 \diamond 1\ 4 \diamond \\
3\ 2\ 5\ 1\ 4 \diamond \\
3\ 2\ 5\ 1\ 4\ 6
\end{array}$$

Figure 5.1: The evolution of the permutation 325146.

The slots of a configuration are labeled from left to right, starting with 1. At each step of the evolution, an insertion is determined by which slot it is inserted into, and how it is inserted. There are four ways to insert a new maximum element n into a slot:

$\diamond \mapsto \diamond n \diamond$	represented by **m** (for middle)
$\diamond \mapsto n \diamond$	represented by **l** (for left)
$\diamond \mapsto \diamond n$	represented by **r** (for right)
$\diamond \mapsto n$	represented by **f** (for fill)

By subscripting the insertion type by which slot we insert into, this encoding is then unique on the set of all permutations. For example, the permutation 325146 considered in Figure 5.1 has the encoding $\mathbf{m}_1\mathbf{m}_1\mathbf{f}_1\mathbf{l}_2\mathbf{f}_1\mathbf{f}_1$. This is called the *insertion encoding*.

For a permutation class C, let $\mathcal{L}(C)$ be the language that is formed by the insertion encodings of the permutations in C. We are going to look particularly at the case when this language is regular.

The prefix p of a word in $\mathcal{L}(C)$ corresponds to some configuration. If it has k slots then, as slots are viewed as promises, the shortest word in $\mathcal{L}(C)$ with this prefix will be of length $|p| + k$ by choosing to fill the promises in some order. We say a configuration is *acceptable* if it corresponds to a prefix of some word in $\mathcal{L}(C)$.

For a language to be regular, it must be accepted by some deterministic finite automata (DFA). In particular, if this DFA has k states, then for any prefix p of any word in $\mathcal{L}$ there exists a word w of length at most $k + |p|$ since we can always choose a path that does not revisit a state. In particular, there is a maximum number of slots allowed on any configuration which can be completed to form a word if $\mathcal{L}$ is regular. The work of Albert, Linton, and Ruškuc [6] tells us that this condition is sufficient, and moreover, they gave a linear time check for a basis as to whether or not this language is regular. This result was rephrased nicely by Vatter [55].

Theorem 5.1.1 (Albert, Linton, and Ruškuc [6], Vatter [55]). For a permutation class $C = \mathrm{Av}(\Pi)$ the following are equivalent:

(i) the set Π contains at least one permutation from each of $\mathrm{Av}(132, 312)$, $\mathrm{Av}(213, 231)$, $\mathrm{Av}(123, 3142, 3412)$, and $\mathrm{Av}(321, 2143, 2413)$,

(ii) there exists an integer k such that there are at most k slots in any acceptable configuration for C,

(iii) the language $\mathcal{L}(C)$ is regular.

The remainder of this section will be focused on showing how to find the insertion encoding, or an equivalent formulation, using the Tilescope algorithm for those whose language is regular. We will call these the *regular insertion encodable* permutation classes. Let $\mathcal{L}_p(C)$ be the set of words in $\mathcal{L}(C)$ with prefix p.

Lemma 5.1.2. Let $C = \mathrm{Av}(\Pi)$ be a permutation class and p be a prefix of a word in $\mathcal{L}(C)$, then the words in $\mathcal{L}_p(C)$ correspond to the underlying permutations of the gridded permutations on a tiling.

(sketch). The 1×1 tiling formed by placing an obstruction for each pattern in Π and a point requirement

$$\mathcal{T} = \left((0,0), \left\{\sigma^{(0,0)} \mid \sigma \in \Pi\right\}, \left\{\left\{1^{(0,0)}\right\}\right\}\right),$$

corresponds to the configuration $\diamond$ and all of the gridded permutations in $\mathrm{Grid}(\mathcal{T})$ correspond to all of the non-empty permutations in C. Let $p = p_1 p_2 \cdots p_k$, and $\mathcal{T}_0 = \mathcal{T}$. Then p_i will be one of $\mathbf{f}_j$, $\mathbf{l}_j$, $\mathbf{r}_j$, or $\mathbf{m}_j$ for some integer j. Build $\mathcal{T}_i$ as follows by adding the point to the j^{th} cell, from the left in the top row containing a point requirement, say $c = (x, y)$. For all the obstructions, add 1 to the y-coordinate of each cell, except those which have points in c. If an obstruction $\pi_1^{d_1}\pi_2^{d_2}\cdots\pi_m^{d_m}$ has a point in c and contains the contiguous subword $\pi' = \pi_{s_1}^c \pi_{s_2}^c \cdots \pi_{s_t}^c$ add the $t + 1$ obstructions created by picking some $s' \in \{0, \ldots, s_t\}$ by placing the points in π' in the cell $(x - 1, y + 1)$ if $s_i \leq s'$ else in cell $(x + 1, y + 1)$, and raising the y-coordinate of the remaining cells of π by 1. If $\pi_{s'}^c = 1^c$ then add the obstruction created by removing 1^c from π and raising the y-coordinate of each cell by 1. If $p_i = \mathbf{f}_j$ add the obstructions $1^{(x-1,y+1)}$ and $1^{(x+1,y+1)}$; if $p_i = \mathbf{l}_j$ add the obstruction $1^{(x-1,y+1)}$; or if

$p_i = \mathbf{r}_j$ add the point obstruction $1^{(x+1,y+1)}$. For p iterate to get $\mathcal{T}_p = \mathcal{T}_k$. In Figure 5.2 the tilings for all of the length 1 prefixes are shown for Av(231).

Then $\mathrm{Grid}\big(\mathcal{T}_\mathrm{p}\big)$ is in bijection with $\mathcal{L}_p(C)$, since we are mimicking the action of each letter at each step, keeping track of the avoidance conditions as we go. Theorem 6.3.3 gives a rigorous proof which proves a more general statement. □

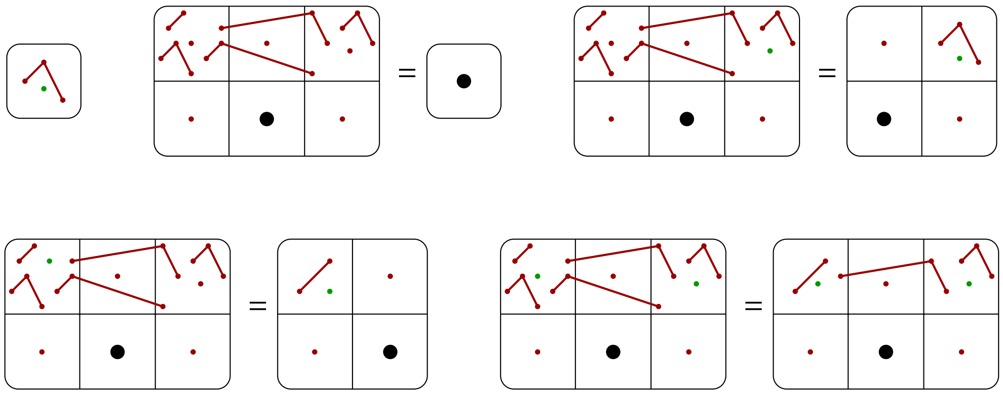


Figure 5.2: The tilings for the prefixes ϵ, $\mathbf{f}_1$, $\mathbf{l}_1$, $\mathbf{r}_1$, and $\mathbf{m}_1$, respectively, as described in Lemma 5.1.2 for Av(231), where for each we have the tiling using the methods of Section 4.2 and removing rows and columns that are empty.

The points in $\mathcal{T}_p$ play no role in terms of avoidance. These points are also placed onto their own row and column, and so we see that $\mathrm{Grid}\big(\mathcal{T}_\mathrm{p}\big)$ can then be enumerated as a Cartesian product of points and the top row. This is a special case of the Factors strategy discussed in Section 4.3. We define the verification strategy PointVerify that only verifies the point tiling.

Given a tiling $\mathcal{T}$, with a row containing j cells which all contain a point requirement. Then there is a length preserving bijection from $\mathrm{Grid}(\mathcal{T})$ to the disjoint union of the tilings created as in the proof of Lemma 5.1.2 by the moves $\mathbf{f}_j, \mathbf{l}_j, \mathbf{r}_j$ and $\mathbf{m}_j$ for all $j \in \{1, 2, \ldots, n\}$. We define InsEnc to be the strategy that takes as input $\mathrm{Grid}(\mathcal{T})$ and returns this combinatorial rule.

After removing the points from $\mathcal{T}_p$ we are left with a $1 \times k$ tiling. By Theorem 5.1.1 there is a bound on the number of slots, and as the non-empty cells in the top row correspond to the slots, this implies there is a bound on the size of k. Each application of InsEnc will add obstructions which are shorter or equal to the length of those in the original tiling. Therefore, there is a bound on the length of obstructions in these tilings, and moreover, this implies there are finitely many $1 \times k$ tilings we will come across.

Theorem 5.1.3. The Tilescope algorithm will terminate on regular insertion encodable permutation classes.

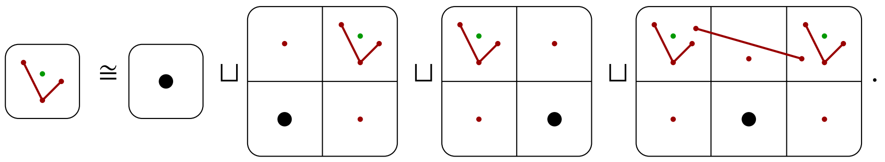
Proof. Use the strategies Factors, InsEnc, and PointVerify. By Theorem 5.1.1 this will terminate, as the top rows in the tilings found must eventually repeat. □

Vatter [55] gave an implementation for finding the regular insertion encoding of permutation classes. This implementation relied on a result that requires generating all permutations from a given configuration to some length. We do not have to do that as the gridded permutations already encode all of the information gained from this. Since the generation of permutations is an expensive operation, the implementation of Theorem 5.1.3 is much faster.

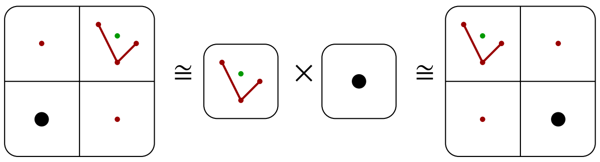
5.2 Row separation

From Theorem 5.1.1(i) it can be seen that Av(312) is not a regular insertion encodable permutation class. It does, however, have a context-free grammar, as given by Albert, Linton, and Ruškuc [6, Proposition 4]. We will rediscover this using gridded permutations.

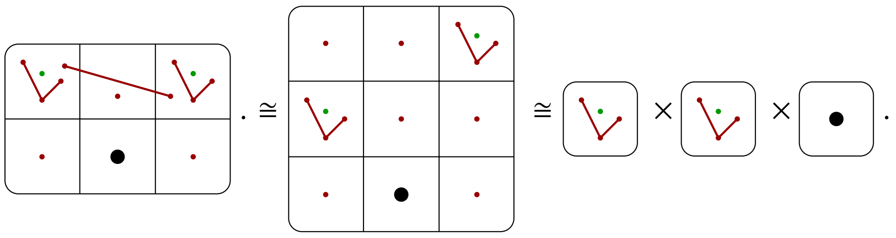
We first apply InsEnc to the tiling corresponding to the prefix ϵ, as in Lemma 5.1.2, and use the methods of Section 4.2 to get that



The first tiling will be verified by PointVerify. Applying Factors will decompose each of the middle two tilings into two 1×1 tilings.



Consider the fourth tiling, corresponding to $\mathcal{T}_{m_1}$. All of the points in cell $(0, 1)$ of a gridded permutation in Grid$(\mathcal{T}_{\mathrm{m}_1})$ will appear below all of the points in cell $(2, 1)$ due to the obstruction $2^{(0,1)}1^{(2,1)}$. Therefore it is equivalent to the tiling where these two cells are put on their own row and column, which we can then apply Factors to.

 (5.1)

Together these combinatorial rules form a combinatorial specification, as depicted in Figure 5.3 Moreover they can be seen to imply the grammar

$$S \mapsto \mathbf{f}_1|\mathbf{l}_1 S|\mathbf{r}_1 S|\mathbf{m}_1 SS,$$

given in Albert, Linton, and Ruškuc [6]. We are now going to define a set of strategies for finding insertion encodings taking into consideration such separations. The combinatorial specifications found can be translated to context-free grammars and so will have algebraic generating functions.

Given a tiling $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R})$ with the obstructiion $2^{(x,y_1)}1^{(x,y_2)}$ we know that for a gridded permutation π in Grid$(\mathcal{T})$ all of the points in (x, y_1) must be below all of the points in (x, y_2). A similar statement holds for the obstruction $1^{(x,y_2)}2^{(x,y_1)}$. If either of these obstructions are in $\mathcal{O}$ we say $(x, y_1) <_{\mathcal{T}} (x, y_2)$. If there is a cell (x, y) that satisfies $(x, y) <_{\mathcal{T}} (x, y')$ for all non-empty cells (x, y') in $\mathcal{T}$, then it is true that all of the points in (x, y) are below the rest of the points in that row. As such we can create the separated tiling as we did in our argument for Av(312) in Equation (5.1).

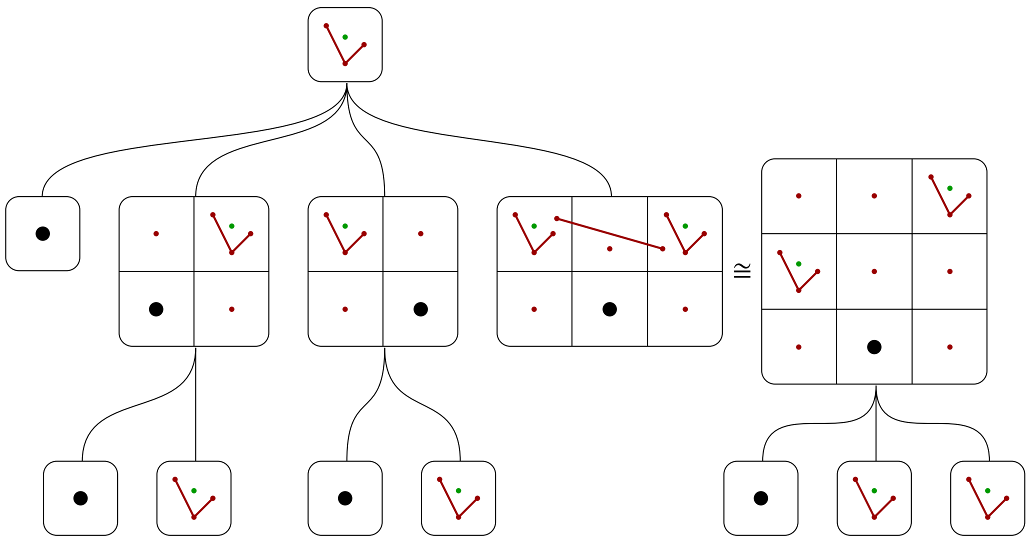
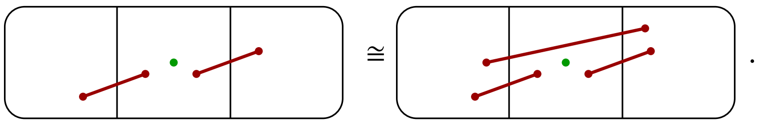


Figure 5.3: A pictorial representation of the combinatorial specification for Av(312) found by Tilescope using the separation insertion encoding.

This idea could also be stated in terms of columns. Therefore, we define RowColSep to be the strategy that takes as input a tiling $\mathcal{T}$ and separates all rows (and columns) maximally with respect to the length two obstructions in $\mathcal{T}$. In practice, what we do is generate all possible separations that satisfy the inequalities and sort these by the number of rows (or columns) we separate into, picking the one with the most separation.

In Section 4.2, we showed that



This essentially says that there is a transitivity of our inequalities through cells containing point requirements. As the InsEnc strategy will leave many point requirements, and more length 2 obstructions can imply more separations, we define the strategy ObsTrans, that adds in the extra length 2 obstructions that can be found in this manner. We collectively call the set of strategies InsEnc, ObsTrans, PointVerify, RowColSep and Factors the *separation insertion encoding*. This can be tried from all directions, and we call this strategy the *separation insertion encoding with symmetries*.

Conjecture 5.2.1. There exists a condition such as the one in Theorem 5.1.1 that charachterizes the bases that will terminate with the separation insertion encoding.

Tenner [53] enumerated the permutation class

$$\mathrm{Av}(4321, 34512, 45123, 35412, 43512, 45132, 45213, 53412, 45312, 45231).$$

Tilescope found a combinatorial specification for this with 100 equations using separation insertion encoding with symmetries. These equations can be solved to give the generating function

$$\frac{1 - 4x + x^3}{1 - 5x + 3x^2 + 2x^3 - x^4}$$

for this permutation class.

In Pantone [50], the permutation classes

$$\text{Av}(3124, 4123, 4231, 4312, 21435, 21534, 32541),$$
$$\text{Av}(3124, 4132, 4312, 21354, 21435, 21543, 31542), \text{ and}$$
$$\text{Av}(3124, 4123, 4132, 4231, 4312, 21354, 21435, 21534, 21543, 31542, 32541)$$

were enumerated using grid classes. Tilescope found combinatorial specifications with 47, 39, and 38 equations, respectively, using separation insertion encoding with symmetries. These solve to give the generating functions

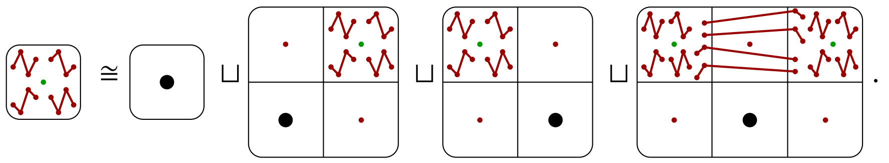
$$\frac{1-5x+7x^2-x^3}{1-6x+11x^2-6x^3}, \quad \frac{1-8x+24x^2-32x^3+19x^4-6x^5+x^6}{1-9x+31x^2-51x^3+41x^4-15x^5+12x^6}, \quad \text{and} \quad \frac{1-6x+13x^2-11x^3+3x^4-2x^5+x^6}{1-7x+18x^2-21x^3+11x^4-2x^5}$$

for each of these permutation classes.

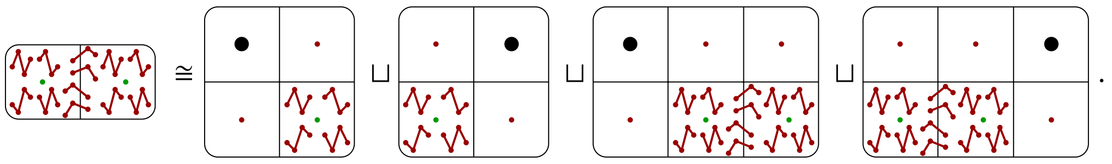
5.3 Allowing more directions

The insertion encoding builds the permutations from bottom to top. Of course, this can be done in any direction: top to bottom, bottom to top, left to right, or right to left. But why not build from both the top and the bottom at the same time? We will consider the permutation class $C = \text{Av}(2143, 2413, 3142, 3412)$ which does not succeed with the regular insertion encoding or with separation insertion encoding.

We will start by applying InsEnc and removing redundant obstructions to get,



After considering the factors, we are left with only the top row of the final tiling to consider. Instead of continuing with InsEnc, we will use the variation where we place the maximum. This would normally return a combinatorial rule with 8 tilings, but we remove the tilings that come from placing in the middle of either cell, the rightmost of cell $(0, 0)$, or the leftmost of cell $(1, 0)$ as each of these choices will lead to always containing at least one of the length 3 obstructions. After removing redundant obstructions from the remaining 4 tilings we have,



After applying Factors, we have a combinatorial specification for Av(2143, 2413, 3142, 3412) as depicted in Figure 5.4. This combinatorial specifiaction leads directly to the generating function

$$\frac{x-2x^2}{1-4x+2x^2}.$$

We define the strategy InsEncTAB that allows inserting both the top or bottom elements of a given row in a tiling and define *separation top and bottom insertion encoding with symmetries* to be the same the set of strategies as *separation insertion encoding with symmetries* except we replace InsEnc with InsEncTAB.

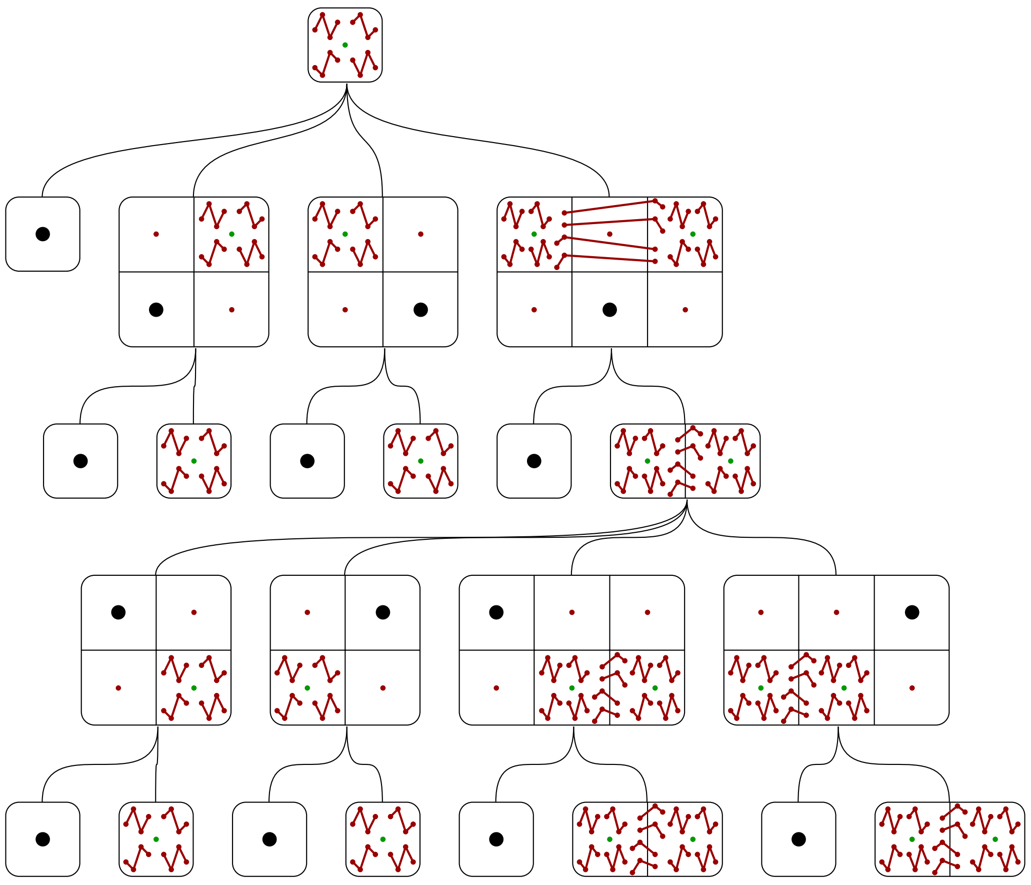


Figure 5.4: A pictorial representation of the combinatorial specification found by Tilescope for Av(2143, 2413, 3142, 3412) using top and bottom insertion encoding.

5.4 Successes

In this section, we will discuss running the Tilescope algorithm with the strategies introduced in this chapter. We will run on all permutation classes whose basis is a subset of $\mathcal{S}_4$. With Theorem 5.1.3 we know Tilescope will terminate on regular insertion encodable permutation classes, and using Theorem 5.1.1 whether a permutation class has a regular insertion encoding can be seen from the basis. Therefore we work on the lexicographically minimal bases which are not regular insertion encodable from any direction. In total there are 2897 such bases, the largest containing 12 patterns.

We ran with three sets of strategies, incrementally increasing the power of the set of strategies. The first set we ran was the separation insertion encoding. The second set was the same, but we tried it from all directions by using the separation insertion encoding with symmetries. The final set of strategies used was the separation top and bottom insertion encoding with symmetries.

If a basis succeeds with the separation insertion encoding, then it will succeed by taking symmetries. If it succeeds with either of those, it will succeed allowing the top and bottom insertions. Therefore we ran one after the other, only running the failures from the previous run. Each basis was run for ten minutes using the separation insertion encoding, then ten minutes with symmetries and finally 60 minutes with the top and bottom strategy. A basic heuristic we came to while running these bases with these strategies, is either they finish reasonably quickly or not at all. The results of these runs can be seen in Table 5.1.

Number of length 4 patterns in basis	Not regular insertion encodable	Success with separation insertion encoding	Success with separation insertion encoding with symmetries	Success with separation top and bottom insertion encoding with symmetries	No success with any of these strategy sets
12	1	1	-	-	0
11	10	8	1	0	1
10	48	34	8	0	6
9	151	93	36	0	22
8	337	171	108	1	57
7	547	211	221	9	106
6	659	178	300	36	145
5	578	94	254	75	155
4	363	26	114	79	144
3	153	2	10	36	105
2	43	0	0	3	40
1	7	0	0	0	7
total	2897	818	1052	239	788

Table 5.1: The successes for bases consisting of length 4 patterns with insertion encoding strategies.

The three 2x4 permutation classes that succeed with separation top and bottom insertion encoding with symmetries are Av(1324, 4231), Av(1342, 2431) and Av(1432, 3412). These combinatorial specifications correspond to systems of 112, 66, and 56 equations, respectively, which can be routinely solved by an algebra system to show that their generating functions are

$$\frac{1-12x+59x^2-152x^3+218x^4-168x^5+58x^6-6x^7}{1-13x+60x^2-202x^3+336x^4-320x^5+160x^6-32x^7}, \quad \frac{1-5x+3x^2+x^2\sqrt{1-4x}}{1-6x+8x^2-4x^3}, \quad \text{and} \quad \frac{1-4x+x^2}{1-5x+4x^2},$$

respectively.

Chapter 6

Point placement

6.1 Placing points on a tiling

In Chapter 5, we introduced strategies that inserted the minimum point in each row and placed this point on a new row and column. In this chapter, we will generalize this idea, although we will move away from the idea of slots. Given a tiling $\mathcal{T}$, we will create a larger tiling with a required point placed into a cell, expanding that row and column, as seen in Figure 6.1. We refer to the tiling with the placed point at cell (i, j) in an $n \times m$ tiling as $\mathcal{T}_{(i,j)}^{(n,m)}$.

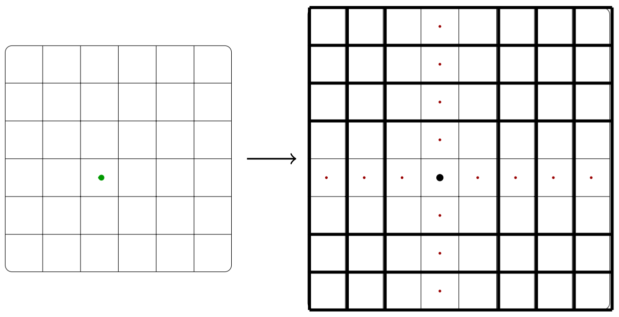


Figure 6.1: On the left is a tiling $\mathcal{T}$ with a point requirement at cell $(2, 2)$. On the right is the tiling $\mathcal{T}_{(2,2)}^{(6,6)}$ with the point requirement of the left tiling placed on a new row and column. The regions enclosed by the bold lines correspond to the cells of $\mathcal{T}$. In Sections 6.2 and 6.3 we discuss how to make this operation a combinatorial rule.

In Figure 6.1 we have drawn bold lines to highlight the regions that the cells from $\mathcal{T}$ will be mapped to. Formally, for an $n \times m$ tiling $\mathcal{T}$ and the cell (i, j) of $\mathcal{T}$ corresponds to the set of cells in $\mathcal{T}_{(i,j)}^{(n,m)}$ given by $\kappa_{(i,j)}^{(n,m)}(a, b) = A \times B$ where

$$A = \begin{cases} \{a\} & a < i \\ \{a, a+1, a+2\} & a = i \\ \{a+2\} & a > i \end{cases} \quad \text{and} \quad B = \begin{cases} \{b\} & b < j \\ \{b, b+1, b+2\} & b = j \\ \{b+2\} & b > j \end{cases}.$$

We want to ensure that the avoidance and containment conditions given by $\mathcal{T}$ are maintained with respect to this expansion of cells.

There are two stages to this process. First, we want the expanded regions to still have the same avoidance and containment conditions as the original tiling. We will cover this in

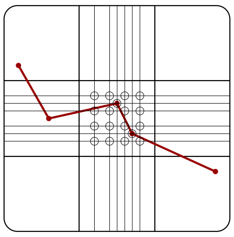


Figure 6.2: A gridded permutation in red, with black circles representing where a placed point in the region could be with respect to the gridded permutation.

Section 6.2. Secondly, we need that the placed point can be uniquely identified from the original tiling to ensure we get a bijection between the sets of gridded permutations defined by these tilings. For example, with the insertion encoding, we uniquely identified the placed point as the lowest point in the given row. We will cover this in Section 6.3.

6.2 Stretching gridded permutations

To place a point (i, π_i) of a gridded permutation $\pi = \pi_1^{c_1}\pi_2^{c_2}\cdots\pi_k^{c_k}$ onto a new row and column we need the following: For integers a, b and x, let

$$\alpha_{a,b}(x) = \begin{cases} x & a < b \\ x+1 & a = b \\ x+2 & a > b \end{cases}.$$

For the point (ℓ, π_ℓ), in cell c_ℓ, we say $\beta_{(a,b)}(\ell, \pi_\ell, c_\ell) = (\alpha_{\ell,a}(c_\ell), \alpha_{\pi_\ell,b}(c_\ell))$. Finally, the mapping $\phi_{(i,\pi_i)}$, given by

$$\phi_{(i,\pi_i)}\left(\pi_1^{c_1}\pi_2^{c_2}\cdots\pi_k^{c_k}\right) = \pi_1^{d_1}\pi_2^{d_2}\cdots\pi_k^{d_k}, \text{ where } d_\ell = \beta_{i,\pi_i}(\ell, \pi_\ell, c_\ell),$$

is the mapping which places the point (i, π_i) onto a new row and column and shifts the remaining points accordingly.

We also need to consider the case where the points of π are not placed on a new row and column. If we are placing the point into cell (i, j) then the point being placed must be on some horizontal line between the points in row j and some vertical line between the points in column i, see Figure 6.2. The points in the column will have r contiguous indices in π, say u, $u+1$, …, $u+r-1$, and the points in the row will have s contigous values v, $v+1$, …, $v+s-1$. We define

$$\begin{aligned} \mathsf{stretch}_{(i,j)}(\pi) &= \{\phi_{(u+n-0.5,v+m-0.5)}(\pi) \mid n \in \{0,1,\ldots,r\} \text{ and } m \in \{0,1,\ldots,s\}\} \\ &\cup \{\phi_{(\ell,\pi_\ell)}(\pi) \mid (\ell,\pi_\ell) \in \pi \text{ such that } c_l = (i,j)\}. \end{aligned}$$

In words, $\mathsf{stretch}_{(i,j)}(\pi)$ is the set of gridded permutations obtained by streching a gridded permutation in a $n \times m$ tiling to a gridded permutation in $\mathrm{Grid}\left(\mathcal{T}_{\mathrm{(i,j)}}^{\mathrm{(n,m)}}\right)$ with the same underlying permutation that maps each gridded position (k, ℓ) to some cell in $\kappa_{(i,j)}^{(n,m)}(k, \ell)$. By construction we get the following.

Theorem 6.2.1. Let σ be a gridded permutation and $\mathcal{T}$ be a tiling. Then $\mathrm{Grid}(\mathcal{T})$ is a subset of $\mathrm{Av}(\sigma)$ if and only if $\mathrm{Grid}\left(\mathcal{T}_{\mathrm{(i,j)}}^{\mathrm{(n,m)}}\right)$ is a subset of $\mathrm{Av}\left(\mathsf{stretch}_{(i,j)}(\sigma)\right)$.

The goal is to find a combinatorial rule for a tiling by placing points onto a new row and column. To do this we will need to find a bijection to a set of tilings. If for a fixed subset $\mathcal{G}' \subseteq \mathcal{G}$ we can select the index of a specific point of each gridded permutation in $\mathcal{G}'$ i.e., some function $f : \mathcal{G}' \mapsto \mathbb{Z}^2$ defined with the mapping $f_n : \mathcal{G}'_n \mapsto \{1, 2, \ldots, n\}$ as $f(\pi) = (f_n(\pi), \pi_{f_n(\pi)})$ e.g., $\mathcal{G}'$ could be the set of all non-empty gridded permutations then $f(\pi)$ might be the topmost point in π, then the mapping $\theta_f : \mathcal{G}' \mapsto \mathcal{G}$ given by $\theta_f(\pi) = \phi_{f(\pi)}(\pi)$ is a well defined map. We call the function f a *point picking function.*

Theorem 6.2.2. For an $n \times m$ tiling $\mathcal{T}$ and a point picking function $f : \mathrm{Grid}(\mathcal{T}) \mapsto \mathbb{Z}^2$ the mapping

$$\theta_f : \mathrm{Grid}(\mathcal{T}) \mapsto \bigsqcup_{(i,\pi_i) \in \{f(\pi) | \pi \in \mathrm{Grid}(\mathcal{T})\}} \mathrm{Grid}\left(\mathcal{T}_{\mathrm{c_i}}^{(\mathrm{n,m})}\right)$$

where c_i is the gridded position of $f(\pi)$ in π is a well-defined mapping.

Proof. Let $f(\pi) = c_i$. Then θ_f maps π to a gridded permutation in $\mathrm{Grid}\left(\mathcal{T}_{\mathrm{c_i}}^{\mathrm{n,m}}\right)$. Since f is well-defined this gridded permutation will always be the same. □

In the next section, we will modify the image of this mapping, for some point picking functions, so that θ_f is a bijection.

6.3 Forced points in requirements

Magnússon [44] introduced forced permutation patterns. The idea is to reduce the number of occurrences of a pattern σ in a permutation π by considering the set of occurrences of σ in π where a particular point is as far in some direction as possible. We extend this definition to gridded permutations.

Definition 6.3.1. Let $\sigma = \sigma_1^{c_1}\sigma_2^{c_2}\cdots\sigma_k^{c_k}$ be a gridded permutation. A *force* F is a point (i, σ_i) with a *direction* from the set $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. For an occurrence of σ in a gridded permutation π, say $\pi_{j_1}^{c_1}\pi_{j_2}^{c_2}\cdots\pi_{j_k}^{c_k}$, the *strength* of the force $F = ((i, \sigma_i), d)$ is

$$\mathrm{strength}\left(\sigma, F, \pi_{j_1}^{c_1}\pi_{j_2}^{c_2}\cdots\pi_{j_k}^{c_k}\right) = \begin{cases} j_i & d = \rightarrow \\ -j_i & d = \leftarrow \\ \pi_i & d = \uparrow \\ -\pi_i & d = \downarrow \end{cases}.$$

Let $\mathrm{occ}_{\sigma,F}(\pi)$ be the set of occurrences of σ in π with maximum strength.

For a gridded permutation σ and force F, the point being forced is uniquely defined within every permutation containing σ. For a tiling $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Co}(\sigma)$ and for every force F on σ define the function $f_{\sigma,F} : \mathrm{Grid}(\mathcal{T}) \mapsto \mathbb{Z}^2$ such that $f_{\sigma,F}(\pi)$ is the unique forced point in π. Then the function $\theta_{f_{\sigma,F}}$ as given in Theorem 6.2.2 is a well-defined map.

Let $\mathcal{T} = ((n, m), \mathcal{O}, \mathcal{R})$ be a tiling such that $\mathcal{R}$ contains the requirement $\{\sigma\}$. For some force $F = ((i, \sigma_i), d)$, let c_i be the gridded position of (i, σ_i) in σ. Define $\mathcal{T}_{\sigma,F}$ to be the tiling $\mathcal{T}_{c_i}^{(n,m)}$ with the additional obstructions and requirements

$$\begin{aligned}
O_{\sigma,F} &= \left\{o = \sigma_{i_1}^{e_1}\sigma_{i_2}^{e_2}\cdots\sigma_{i_k}^{e_k} \in \mathsf{stretch}_{c_i}(\sigma) \mid e_i \text{ is farther in direction } d \text{ than } (i+1, j+1)\right\} \\
&\quad \cup \left(\bigcup_{o\in O} \mathsf{stretch}_{c_i}(o)\right) \\
\mathcal{R}_{\sigma,F} &= \left\{\bigcup_{r\in\mathcal{R}_i} \mathsf{stretch}_{c_i}(r) \mid \mathcal{R}_i \in \mathcal{R}\right\}.
\end{aligned}$$

With this we can refine the image of the mapping $\theta_{\sigma,F}$ to get a bijection.

Theorem 6.3.2. Let σ be a gridded permutation. For a tiling $\mathcal{T} = ((n,m), O, \mathcal{R}) \subseteq \mathrm{Co}(\sigma)$, $(\mathrm{Grid}(\mathcal{T}), \{\mathrm{Grid}(\{\mathcal{T}_{\sigma,\mathrm{F}}\})\}, \sqcup)$ is a combinatorial rule.

Proof. The mapping $\theta_{\sigma,F}$ is a length preserving mapping that also preserves the underlying permutation. We will show it is a bijection.

Let π_1 and π_2 be two gridded permutations in $\mathrm{Grid}(\mathcal{T})$ such that $\theta_{\sigma,F}(\pi_1) = \theta_{\sigma,F}(\pi_2)$, then it follows that $\pi_1 = \pi_2$ since $\theta_{\sigma,F}$ maps the gridded positions with the mapping $\kappa_{(i,j)}^{(n,m)}$.

Let π be a gridded permutation in $\mathrm{Grid}(\mathcal{T}_{\sigma,\mathrm{F}})$. Let π' be the gridded permutation formed by replacing a gridded position c_i in π with the inverse mapping of $\kappa_{(i,j)}^{(n,m)}$. As the gridded permutations in $\mathrm{Grid}(\mathcal{T}_{\sigma,\mathrm{F}})$ avoid all of the stretched gridded permutations in $\mathsf{stretch}_{(i,j)}(\sigma)$ which are farther in direction d it is clear that $\theta_{\sigma,F}(\pi')$ will place the unique forced point in π in cell $(i+1, j+1)$, and therfore $\theta_{\sigma,F}(\pi') = \pi$. □

Theorem 6.2.1 shows that if $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Av}(\sigma)$ then $\mathrm{Grid}(\mathcal{T}_{\sigma,\mathrm{F}}) \subseteq \mathrm{Av}(\sigma)$. Also, for a set of gridded permutations $\mathcal{R}$, Theorem 6.2.1 shows that if $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Co}(\mathcal{R})$ then $\mathrm{Grid}(\mathcal{T}_{\sigma,\mathrm{F}}) \subseteq \mathrm{Co}(\mathcal{R})$. We define PointPlacement to be the equivalence strategy which returns the combinatorial rules in Theorem 6.3.2.

If a tiling $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Co}(\mathcal{R})$, then we could also find the point which is the lowest in an occurrence of some $\sigma \in \mathcal{R}$. We will do this one level of generality further, that is we will consider $\mathrm{Grid}(\mathcal{T}) \subseteq \mathrm{Co}(\mathcal{R}_1) \cap \mathrm{Co}(\mathcal{R}_2) \cap \cdots \cap \mathrm{Co}(\mathcal{R}_k)$. For a direction $d \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ let the mapping $f_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)} : \mathrm{Co}(\mathcal{R}_1), \mathrm{Co}(\mathcal{R}_2) \cap \cdots \cap \mathrm{Co}(\mathcal{R}_k) \mapsto \mathbb{Z}^2$ be defined by: $f_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)}$ is the point furthest in the direction d in π that is in an occurrence of a pattern σ in some $\mathcal{R}_i$.

For each (i,j) in $\{f_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)} \mid \pi \in \mathrm{Grid}(\mathcal{T})\}$ define the tiling $\mathcal{T}_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)}^{(i,j)}$ to be the tiling $\mathcal{T}_{(n,m)}^{(i,j)}$ with the additional obstruction and requirements

$$\begin{aligned}
O_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)}^{(i,j)} &= \left(\bigcup_{o\in O} \mathsf{stretch}_{(i,j)}(o)\right) \\
&\quad \cup \{o = \sigma_{i_1}^{e_1}\sigma_{i_2}^{e_2}\cdots\sigma_{i_k}^{e_k} \in \mathsf{stretch}_{i,j}(\sigma) \mid \sigma \in \bigcup_{i=1}^{k} \mathcal{R}_i \text{ and } e \text{ is} \\
&\qquad \text{farther in direction } d \text{ than } (i+1, j+1)\} \\
\mathcal{R}_{d,(\mathcal{R}_1,\mathcal{R}_2,\ldots,\mathcal{R}_k)}^{(i,j)} &= \left\{\bigcup_{r\in\mathcal{R}_i} \mathsf{stretch}_{(i,j)}(r) \mid \mathcal{R}_i \in \mathcal{R}\right\}.
\end{aligned}$$

Theorem 6.3.3. Let $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k$ be sets of gridded permutations. Let $\mathcal{T} = ((n,m), \mathcal{O}, \mathcal{R})$ be a tiling that is a subset of each $\mathrm{Co}(\mathcal{R}_i)$, then

$$\left(\mathrm{Grid}(\mathcal{T}), \left\{\mathrm{Grid}\left(\mathcal{T}^{\mathrm{(i,j)}}_{\mathrm{d},(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_\mathrm{k})}\right) \mid (i,j) = f_{d,(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_k)}(\pi) \text{ for some } \pi \in \mathrm{Grid}(\mathcal{T})\right\}, \sqcup\right)$$

is a combinatorial rule.

Proof. The mapping $\theta_{f_{d,(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_k)}}$ is a length preserving mapping as it preserves the underlying permutation. We will show it is a bijection.

Let π_1 and π_2 be two gridded permutations in $\mathrm{Grid}(\mathcal{T})$ such that

$$\theta_{f_{d,(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_k)}}(\pi_1) = \theta_{f_{d,(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_k)}}(\pi_2)$$

is in the same tiling $\mathrm{Grid}\left(\mathcal{T}^{\mathrm{(i,j)}}_{\mathrm{d},(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_\mathrm{k})}\right)$, then it follows that $\pi_1 = \pi_2$ since $\theta_{f_{d,(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_k)}}$ maps the gridded positions with the mapping $\kappa^{(n,m)}_{(i,j)}$.

Let π be a gridded permutation in $\mathrm{Grid}\left(\mathcal{T}^{\mathrm{(i,j)}}_{\mathrm{d},(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_\mathrm{k})}\right)$. Let π' be the gridded permutation formed by replacing a gridded position c_i in π with the inverse mapping of $\kappa^{(n,m)}_{(i,j)}$. As gridded permutations in $\mathrm{Grid}\left(\mathcal{T}^{\mathrm{(i,j)}}_{\mathrm{d},(\mathcal{R}_1,\mathcal{R}_2,\dots,\mathcal{R}_\mathrm{k})}\right)$ avoids all of the stretched gridded permutations in $\mathsf{stretch}_{(i,j)}(r)$ for every gridded permutation r in some $\mathcal{R}_i$ which are farther in direction d it is clear that $\theta_{\sigma,F}(\pi')$ will place the unique forced point in π in cell $(i+1, j+1)$, and therefore $\theta_{\sigma,F}(\pi') = \pi$. □

As before this mapping also preserves the avoidance and containment conditions. We define the strategy ExtremePointPlacement(d) that returns the combinatorial rules in direction d from Theorem 6.3.3.

The InsEnc fits into this framework by applying the ExtremePointPlacement(↓) strategy to the set of singleton requirements with the length 1 gridded permutation in each cell corresponding to a slot. To capture the InsEnc precisely some work needs to be done to ensure the cells are corresponding to slots in the end, but with the ideas in Section 6.4 InsEnc is recovered.

6.4 Requirement insertion and placement

The most basic thing we can say when looking for the structure of permutations is either a permutation is empty or it is not. Said differently, either a permutation contains 1 or avoids 1. This statement is true for any pattern and can be stated for gridded permutations also.

Theorem 6.4.1. Let $\mathcal{T} = ((n,m), \mathcal{O}, \mathcal{R}))$ be a tiling and σ be a gridded permutation. Define the tilings

$$\mathcal{T}_{\mathrm{Av}(\sigma)} = ((n,m), \mathcal{O} \cup \{\sigma\}, \mathcal{R}) \quad \text{and} \quad \mathcal{T}_{\mathrm{Co}(\sigma)} = ((n,m), \mathcal{O}, \mathcal{R} \cup \{\{\sigma\}\}),$$

then $(\mathrm{Grid}(\mathcal{T}), \{\mathrm{Grid}(\mathcal{T}_{\mathrm{Av}(\sigma)}), \mathrm{Grid}(\mathcal{T}_{\mathrm{Co}(\sigma)})\}, \sqcup)$ is a combinatorial rule.

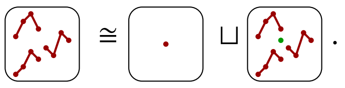
Proof. Every gridded permutation either avoids or contains σ, so

$$\mathrm{Grid}(\mathcal{T}) = (\mathrm{Grid}(\mathcal{T}) \cap \mathrm{Av}(\sigma)) \sqcup (\mathrm{Grid}(\mathcal{T}) \cap \mathrm{Co}(\sigma)) = \mathrm{Grid}(\mathcal{T}_{\mathrm{Av}(\sigma)}) \sqcup \mathrm{Grid}(\mathcal{T}_{\mathrm{Co}(\sigma)}).$$ □

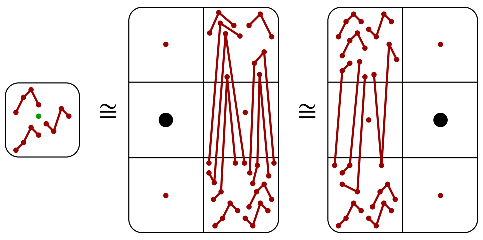
Given this theorem we define ReqInsertion(k) to be the strategy that returns these combinatorial rules for local gridded permutations of length at most k. For example, ReqInsertion(1) says either a cell is empty, or it is not.

We will refer to the set of strategies ReqInsertion(k), PointPlacement, RowColSep, ObsTrans, Factors, LocallyFactorable as *length k requirement insertion with point placement*. The remainder of this section will give an example of a combinatorial specification found using length 2 requirement insertion with point placement for Av(1243, 1342, 2143), first enumerated by Mansour and Shattuck [48] (it is T_6 in their paper).

A permutation π in Av(1243, 1342, 2143) is either empty or contains at least one point, i.e., performing ReqInsertion(1) gives

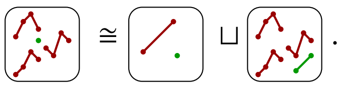


We can then place the point requirement we have introduced using PointPlacement. We have shown the leftmost and rightmost cases, hence these are two representations of non-empty permutations in Av(1243, 1342, 2143).

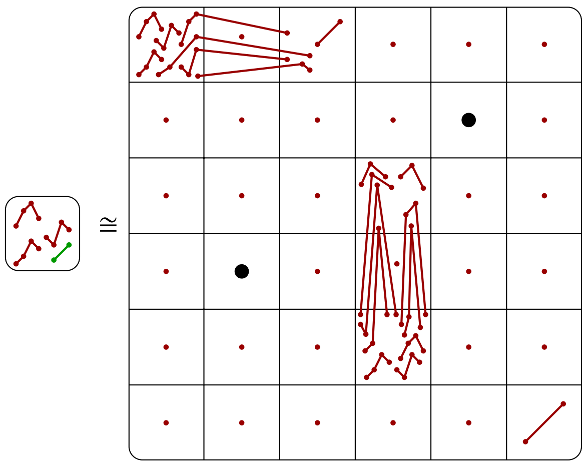


(6.1)

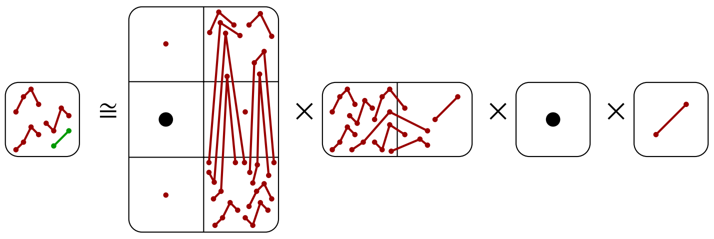
Alternatively, we can say a non-empty permutation in Av(1243, 1342, 2143) is either a non-empty decreasing permutation or contains 12, i.e., performing ReqInsertion(2) gives



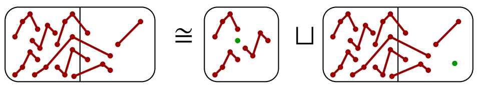
If we apply PointPlacement to the second tiling twice, first placing the point with force $((2,2), \rightarrow)$ and then placing the point with the force $((1,1), \leftarrow)$, after applying RowColSep, we will get the following combinatorial rule,



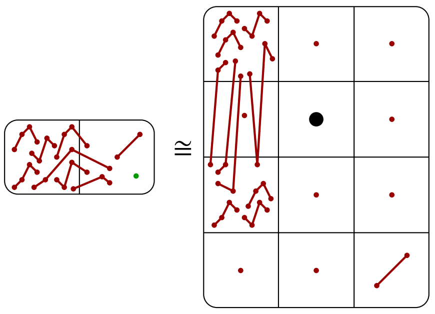
which we can apply Factors to, giving the combinatorial rule



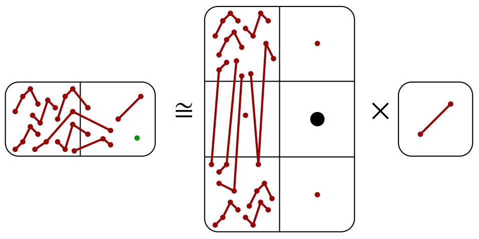
where the first is equivalent to any non-empty permutation in the permutation class, see Equation (6.1). We then only need to explore the second tiling on the right hand side. In this tiling, either cell $(1, 0)$ is empty or contains a point.



If we place this point with force $((1, 1), \leftarrow)$ we get



which applying Factors to gives



where the first tiling on the right is equivalent to a non-empty permutation in the permutation class, see Equation (6.1). Thus we have described a combinatorial specification for Av(1243, 1342, 2143), as depicted in Figure 6.3, which routinely gives the generating function

$$\frac{1 + x - \sqrt{1 - 6x + 5x^2}}{4x - 2x^2}$$

for the binomial transform of Fine's sequence, A033321 on the OEIS.

6.5 Elementary combinatorial specifications for permutation classes

For a regular insertion encodable permutation class $\mathrm{Av}(\Pi)$ it can be seen from Theorem 5.1.1 that for an arbitrary pattern σ the permutation class $\mathrm{Av}(\Pi \cup \{\sigma\})$ is also regular insertion

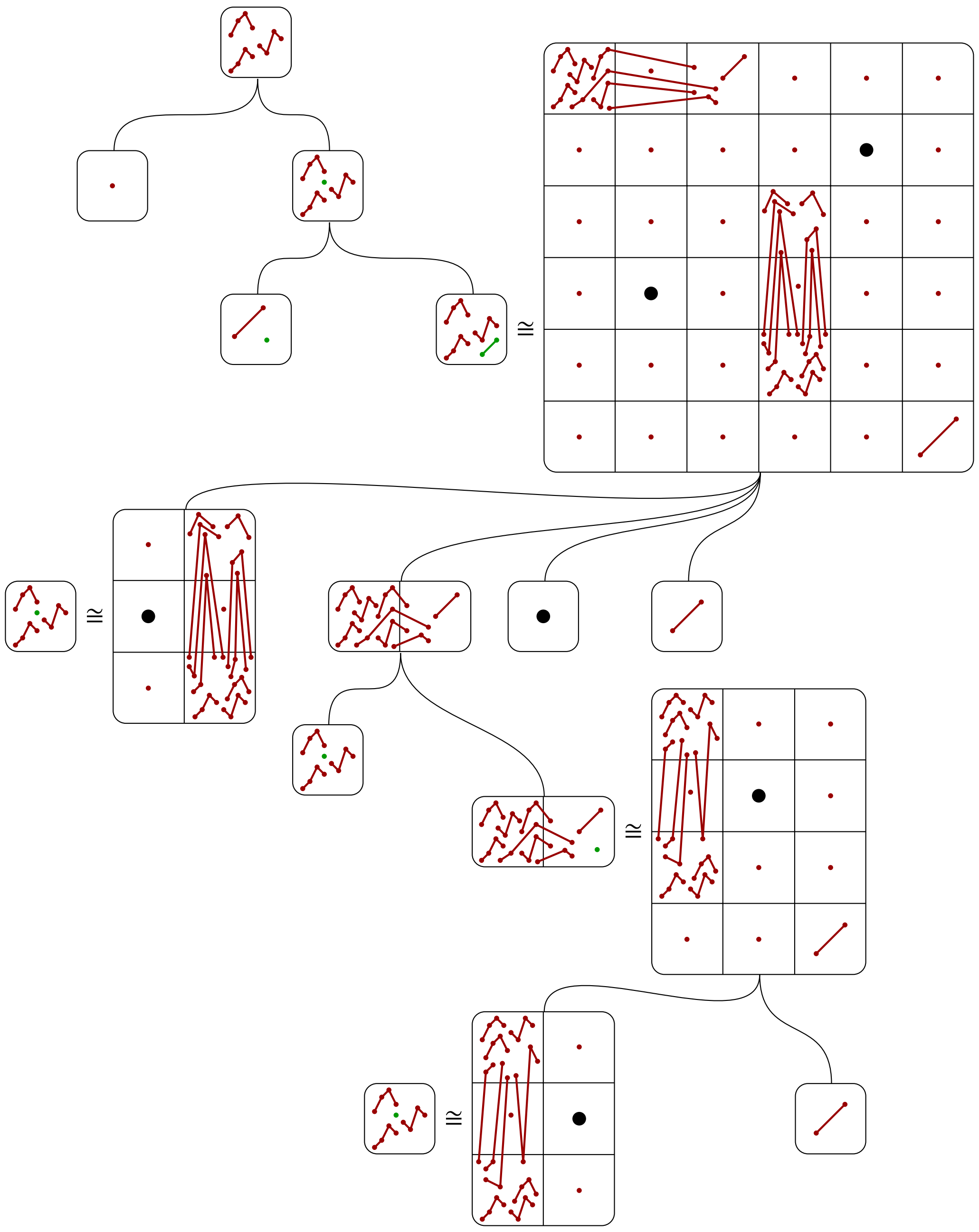
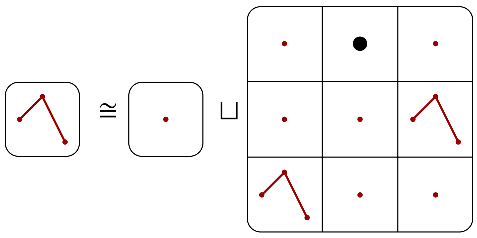


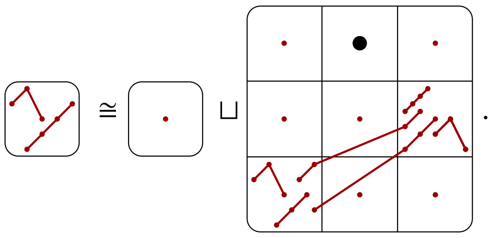
Figure 6.3: A pictorial representation of the combinatorial specification found by Tilescope for Av(1243, 1342, 2143).

encodable. In this section, we will find a similar condition for some of the combinatorial specifications the Tilescope algorithm returns.

As we have seen a few times in this thesis, Av(231) can be described by the combinatorial rule



which leads to the Catalan generating function. This would be found using the strategies ReqInsertion(1), PointPlacement, RowColSep, Factors and PointVerify. For a permutation σ, if we enforce that each tiling in this combinatorial rule avoids σ this will be a combinatorial rule for Av$(231, \sigma)$. For example, if $\sigma = 1234$, we would get

 (6.2)

After adding the avoidance condition for an arbitrary σ second tiling in this combinatorial rule will always be locally factorable and can be enumerated using the methods in Section 4.3.

The reason we could add an arbitrary permutation σ and get back a combinatorial rule that is easy to enumerate is that each tiling in the rule is *fully separated* i.e., there are no non-empty cells in the same row or column. Given such a tiling, all obstructions must be local or locally factorable, and in particular, verified by the LocallyFactorable strategy. We will call such a tiling *elementary*, and define ElementaryVerify to be the verification strategy that verifies all elementary tilings.

We define a permutation class C to be *elementary* if it can be represented as a non-trivial disjoint union of elementary tilings in such a way that the underlying permutations of the gridded permutations in the tilings are precisely the permutations in C. We say the disjoint union of tilings is trivial if it consists of a single tiling, or if all of the tilings are 1.

Theorem 6.5.1. Let $C = \text{Av}(\Pi)$ be an elementary permutation class, given by

$$C = S(\mathcal{T}_1) \sqcup S(\mathcal{T}_2) \sqcup \cdots S(\mathcal{T}_k)$$

where $S(\mathcal{T})$ is the set of underlying permutations of the gridded permutations in Grid$(\mathcal{T})$. For any pattern σ in $\mathcal{S}$ the permutation class $\text{Av}(\Pi \cup \{\sigma\})$ is elementary if there is no $\mathcal{T}_i$ with $\text{Av}(\Pi \cup \{\sigma\}) \subseteq S(\mathcal{T}_i)$.

Proof. Let $\mathcal{T}_i{}'$ be the same as tiling $\mathcal{T}_i$ that additionally avoids all gridded permutations in $\mathcal{G}^{(n,m)}$ with underlying permutation σ, then $S(\mathcal{T}_i{}') \subseteq \text{Av}(\sigma)$. Each of the gridded permutations we have added will be locally factorable because $\mathcal{T}_i$ was assumed to be fully separated. Therefore,

$$C \cap \text{Av}(\sigma) = S(\mathcal{T}_1{}') \sqcup S(\mathcal{T}_2{}') \sqcup \cdots \sqcup S(\mathcal{T}_k{}')$$

where each $\mathcal{T}_i'$ is an elementary tiling. Since there are at least two tilings in the original disjoint union of sets which contain elements from $\mathrm{Av}(\Pi \cup \{\sigma\})$ this will be a non-trivial disjoint union. □

The definition of an elementary permutation class can be restated in a way more akin to peg permutations defined in Section 2.4. Define a *generalized peg permutation* to be a permutation where each letter is decorated with a permutation set of the form

$$\mathrm{Av}(\Pi) \cap \bigcap_{\sigma \in \Pi'} \mathrm{Co}(\sigma)$$

for two permutation sets Π and Π'. Let M_ρ be the matrix defined by $M_{i,j} = S$ if $\rho_i = j^S$ that is $\rho - i$ is decorated with the set S, then $\mathrm{Grid}(\rho) = \mathrm{Grid}\big(\mathrm{M}_\rho\big)$ as for peg permutations. We say a generalized peg permutation ρ is *valid* if each permuation in $\mathrm{Grid}(\rho)$ has a unique gridding.

Theorem 6.5.2. A permutation class is elementary if there exists a finite set of valid generalized peg permutations H such that

$$C = \bigsqcup_{\rho \in H} \mathrm{Grid}(\rho)\,.$$

Proof. After repeated application of ReqInsertion, any elementary tiling can be given by a disjoint union of tilings with local obstructions and requirements. These correspond directly to valid generalized peg permutations. □

For example, in Section 4.3 this procedure was applied to the tiling representing non-empty permutations in $\mathrm{Av}(231, 1234)$ in Equation (6.2).

Corollary 6.5.3. All polynomial permutation classes are elementary.

Proof. This follows from Theorem 6.5.2 and Theorem 2.4.2. □

As evidenced by $\mathrm{Av}(231)$ not all elementary permutation classes are polynomial. A permutation class is elementary if there exists an iterative combinatorial specification where each of the terminal states is an elementary tiling. We search for such specifications for all permutation classes with basis $\Pi \subseteq \mathcal{S}_4$ using the sets of strategies

- ReqInsertion(k), PointPlacement, Factors, ObsTrans, RowColSep, and ElementaryVerify (for $k \in \{1, 2, 3\}$)
- InsEnc, Factors, ObsTrans, RowColSep, and ElementaryVerify
- InsEncTAB, Factors, ObsTrans, RowColSep, and ElementaryVerify

for 24 hours each. From Theorem 6.5.1, if we find an elementary iterative combinatorial specification for a permutation class then by adding another length 4 pattern to the basis this will remain elementary. Therefore we first ran bases with one pattern, then two patterns, etc., removing those which have a shorter elementary specification found by the algorithm that implies it is elementary. The results are given in Table 6.1.

There are 3 polynomial permutation classes with two length 4 patterns, but despite Corollary 6.5.3 the Tilescope algorithm was unable to find a witnessing elementary combinatorial specification for two of these, only succeeding for the finite class $\mathrm{Av}(1234, 4321)$.

$\|\Pi\|$	non-symmetric	minimal elementary bases	total number of elementary bases	total number of non-elementary bases	non-insertion-encodable and non-elementary
12	342424	0	342424	0	0
11	316950	0	316949	1	0
10	249624	0	249611	13	0
9	166786	0	166717	69	0
8	94427	0	94196	231	3
7	44767	0	44260	507	28
6	17728	5	16933	795	108
5	5733	44	4890	843	222
4	1524	334	903	621	244
3	317	38	44	273	143
2	56	1	1	55	43
1	7	0	0	7	7
total	-	422	-	3416	798

Table 6.1: The successes for bases consisting of length 4 patterns with elementary point placement strategies.

6.6 Successes

We are now going to turn the algorithm to full strength on the bases which are non-elementary and non-regular-insertion-encodable. We searched for recursive combinatorial specification for each permutation class with the sets of strategies

- ReqInsertion(k), PointPlacement, Factors, ObsTrans, RowColSep, and LocallyFactorable (for $k \in \{1, 2, 3\}$)
- InsEnc, Factors, ObsTrans, RowColSep, and LocallyFactorable
- InsEncTAB, Factors, ObsTrans, RowColSep, and LocallyFactorable

first for 1 hour, then 24 hours and finally 14 days. The results of these runs are given in Table 6.2. The longest time taken to successfully find a combinatorial specification was approximately 4 days for Av(1234, 1324, 3412).

The successful bases with two length four patterns are listed in Table 6.3. Some other notable successes outside of $\mathcal{S}_4$ include Av(4231, 35142, 42513, 351624) which correspond to the DBI Schubert variety and was first enumerated by Albert and Brignall [5]. Tilescope found a combintatorial specification with 50 equations that can be solved to find the generating function

$$\frac{3 - 17x + 8x^2 - \left(1 - 5x - 2x^2\right)\sqrt{1 - 4x}}{2\left(1 - 6x + 5x^2 - 4x^3\right)}.$$

Brignall and Sliačan [23] enumerated Av(4132, 4231, 31254, 41253) that can be equivalently described as the set of permutations that are juxtapositions of the form $\alpha\beta$ where $\alpha \in \text{Av}(312)$ and $\beta \in \text{Av}(21)$. Tilescope found a combinatorial specification with 37

$\lvert\Pi\rvert$	non-insertion-encodable and non-elementary	success after 1 hour	success after 24 hours	success after 14 days	number of bases remaining
8	3	2	1	0	0
7	28	20	6	0	2
6	108	82	14	0	12
5	222	172	20	0	30
4	244	180	21	1	42
3	143	80	17	7	39
2	43	9	5	0	29
1	7	0	0	0	7
total	798	545	84	8	161

Table 6.2: The successes for bases consisting of length 4 patterns with point placement strategies.

equations that lead to the generating function

$$\frac{\left(1-6x+7x^2+\left(1-4x+x^2\right)\sqrt{1-4x}\right)\sqrt{1-x}+\left(1-4x+3x^2+\left(1-2x+x^2\right)\sqrt{1-4x}\right)\sqrt{1-5x}}{\left(1-7x+12x^2-6x^3+\left(1-5x+4x^2\right)\sqrt{1-4x}\right)\sqrt{1-x}+\left(1-5x+6x^2-2x^3+\left(1-3x+2x^2\right)\sqrt{1-4x}\right)\sqrt{1-5x}}$$

for this permutation class.

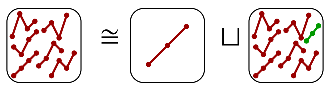
Π	number of equations	OEIS sequence	reference to first enumeration
1243, 2143	65	A155069	Kremer [39, 40]
1243, 2413	12	A165538	Albert, Atkinson, and Vatter [4]
1324, 2143	71	A032351	Bóna [20]
1324, 2413	10	A032351	Bóna [20]
1324, 4231	112	A165528	Albert, Atkinson, and Vatter [3]
1342, 2413	9	A165541	Albert, Atkinson, and Vatter [4]
1342, 2143	20	A109033	Le [42]
1342, 2413	15	A165541	Albert, Atkinson, and Vatter [4]
1342, 2431	12	A032351	Bóna [20]
1342, 3142	6	A155069	Kremer [39, 40]
1342, 3241	24	A032351	Bóna [20]
1432, 2413	12	A032351	Bóna [20]
1432, 3412	15	A047849	Kremer and Shiu [41]
2413, 3142	14	A155069	Kremer [39, 40]

Table 6.3: Bases consisting of two length 4 patterns that succeed with point placement strategies. They are listed with the number of equations the combinatorial specification has, their OEIS sequence, and the reference to the paper that first enumerated it.

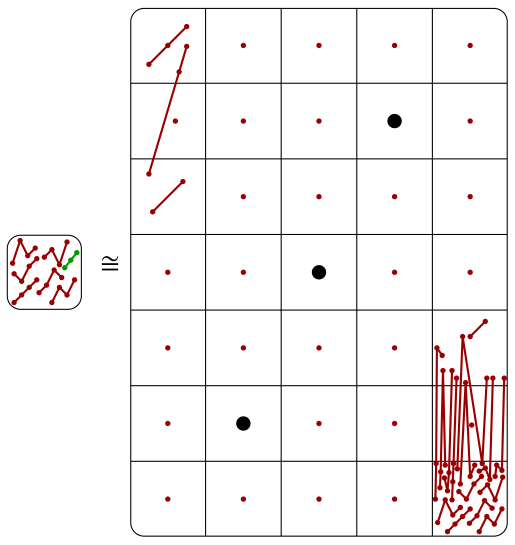
6.7 Combinatorial systems

To illustrate an extension of the Tilescope algorithm we will review the first proof of Theorem 5 from Claesson [27], which enumerates $\mathcal{C} = \text{Av}(1243, 1234, 1324, 1423, 2134, 2314)$, one of the permutation classes Tilescope failed on. We will translate this proof into combinatorial rules on tilings found by our strategies. This permutation class is known as the 123-*segmented permutations* and can alternatively be described as the permutations that avoid a non-consecutive occurrence of 123.

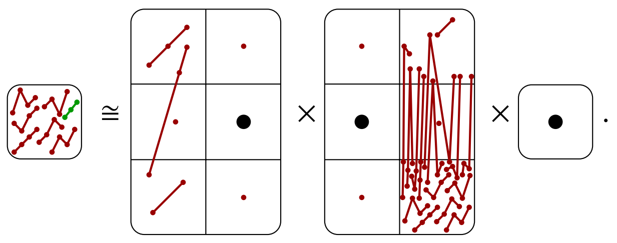
Every permutation in the basis of $\mathcal{C}$ contains 123. Therefore every permutation in the class is either in Av(123) or contains 123, giving the combinatorial rule



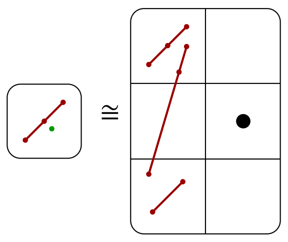
found by ReqInsertion(3). Every permutation in $\mathcal{C}$ containing 123 must contain a leftmost occurrence of 123, that is place the points with forces $((1,1), \leftarrow)$, $((2,2), \leftarrow)$ and $((3,3), \leftarrow)$ to get the combinatorial rule



which after applying Factors gives the combinatorial rule



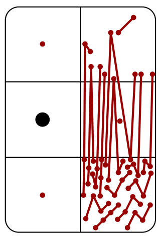
The first tiling on the right hand side represents any non-empty permutation in Av(123) since



by using PointPlacement with the force $((1, 1), \rightarrow)$, i.e. placing the rightmost point. If we let $B(x)$ be the generating function for the Catalan numbers and $F(x)$ be the generating function for C, then from our combinatorial rules $F(x)$, satisfies the equation

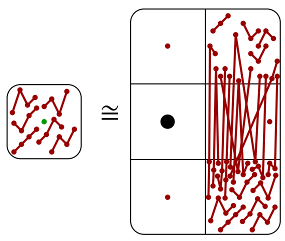
$$F(x) = B(x) + x(B(x) - 1)A(x) \tag{6.3}$$

where $A(x)$ is the generating function for the tiling



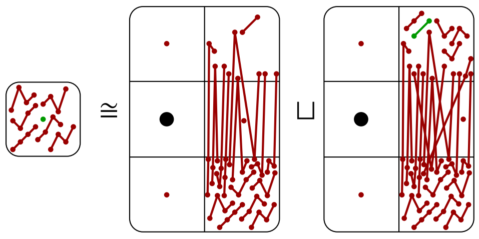
.

We now consider, every permutation in C is either empty or not. If we consider placing the leftmost point of a non-empty permutation then we get the combinatorial rule



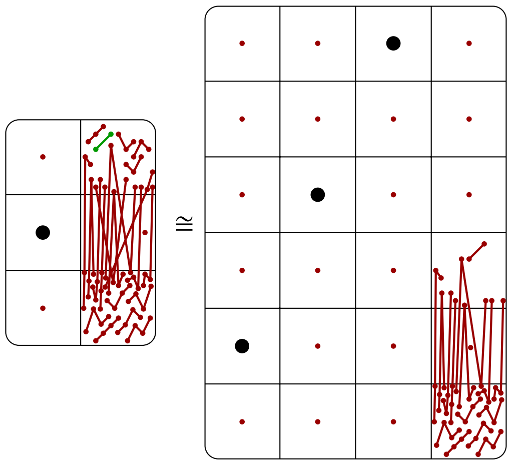
.

The top cell either avoids 12 or contains 12 giving the combinatorial rule

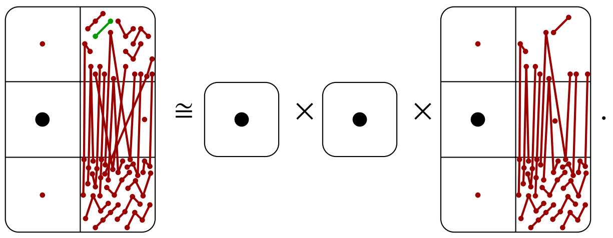


.

If we place the points in the 12 with forces $((1, 1), \leftarrow)$ and $((2, 2), \leftarrow)$ then we get the combinatorial rule



which after applying Factors gives



Therefore $F(x)$ also satisfies the equation

$$F(x) = 1 + A(x) + x^2 A(x). \tag{6.4}$$

Solving equations (6.3) and (6.4) for $F(x)$ and eliminating $A(x)$ gives the generating function

$$\frac{1 - x + 3x^2 - (1 - x + x^2)\sqrt{1 - 4x}}{x + 2x^2 + 2x^3 + x\sqrt{1 - 4x}}$$

for C.

What we have described here is not a combinatorial specification as there are multiple combinatorial rules with the same left-hand side. Despite this, the system of combinatorial rules captures the enumerative information. We define a *combinatorial system* to be any set of combinatorial rules. The exact theory of when a combinatorial system gives the enumeration is unknown to us at the time of writing. Therefore our approach is somewhat experimental.

Using these methods Tilescope has been able to find the enumeration of many other permutation classes by finding combinatorial systems. In particular, it has been able to bring down the number of bases consisting of length four patterns remaining in Table 6.2 from 161 down to 77. The largest basis that remains unenumerated has five length four patterns. The automatic methods in this thesis can therefore enumerate all of the cases in Mansour and Schork [47], Mansour and Schork [45], and Mansour and Schork [46]. Details of the results can be seen in Table 6.4.

The Tilescope algorithm generates a set of combinatorial rules. Each rule corresponds to an equation satisfied by the generating function of a set of permutations that is a subset of the permutation class we are trying to enumerate. Within this system of equations we then attempt to eliminate, as we did to solve equations (6.3) and (6.4), and hopefully find a solution for the generating function of the permutation class in question.

However, as of writing, this method has only been semi-automated. Our current approach is as follows, first, run the Tilescope algorithm for a set amount of time, and then collect the system of equations. Input these to a separate program in Maple that then tries to solve for the generating function. As this method is still in its early days for us, we anticipate that it will enumerate even more permutation classes when developed further.

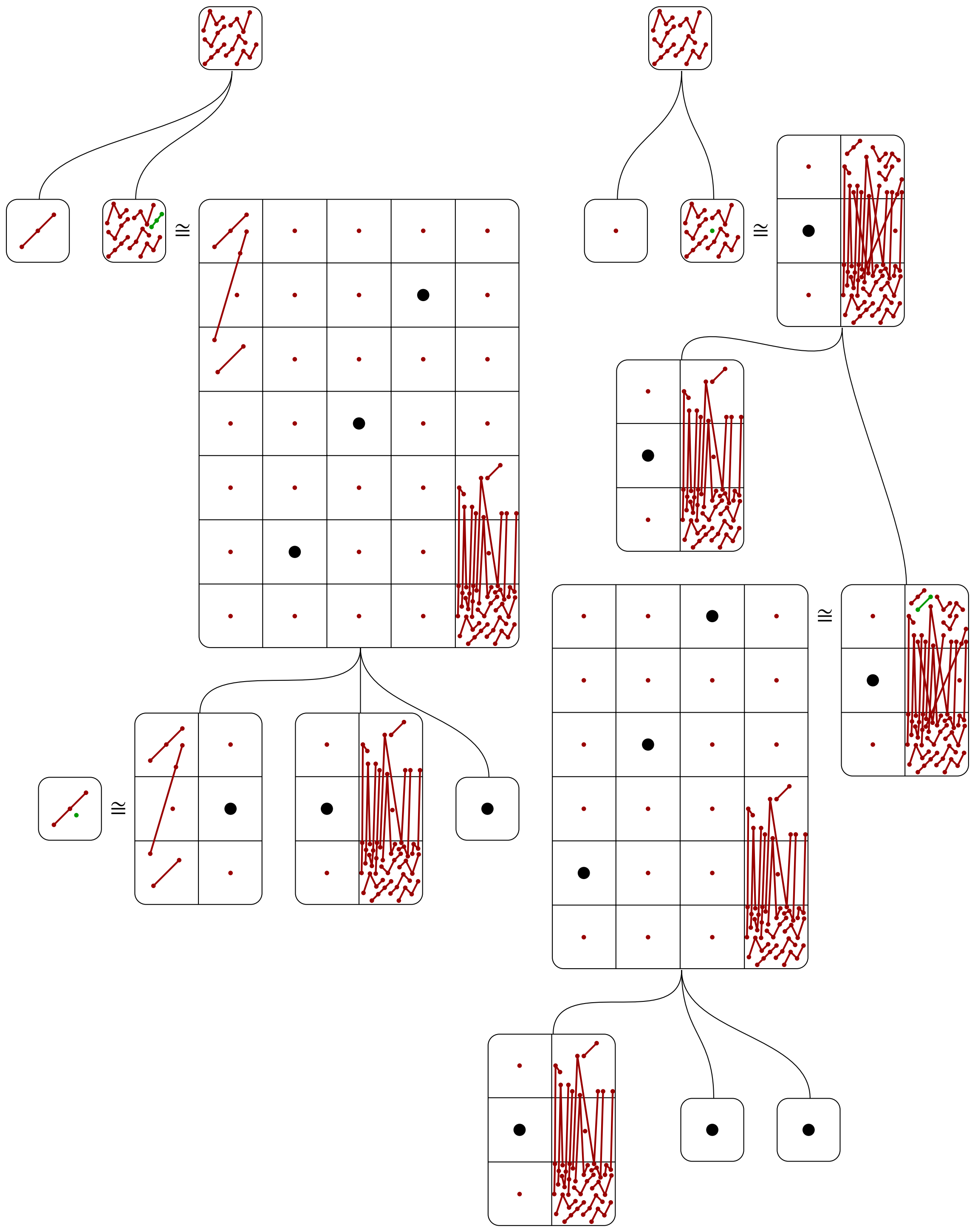


Figure 6.4: A pictorial representation of the combinatorial system found by Tilescope for Av(1243, 1234, 1324, 1423, 2134, 2314).

$\|\Pi\|$	no combinatorial specification	found enumerable combintorial system	number of bases remaining
7	2	2	0
6	12	12	0
5	30	28	2
4	42	31	11
3	39	8	31
2	29	2	27
1	7	0	7
total	161	84	77

Table 6.4: The successes for bases consisting of length 4 patterns with combinatorial systems using point placement strategies.

Chapter 7

Conclusion

7.1 Permutations avoiding 123

As was pointed out in Section 1.2, permutation patterns can be dated back to 1915 when MacMahon [43] showed that Av(123) is enumerated by the Catalan numbers. For this reason, it is very frustrating that all of the automatic methods introduced in this thesis have not found a combinatorial specification for Av(123). In this section, we discuss two methods for enumerating Av(123) which are not yet automated. If automated, one would anticipate they will be capable of enumerating many other permutation classes.

7.1.1 Left-to-right minima

The arguments in this subsection are from Bean, Claesson, and Ulfarsson [15].

Given a permutation avoiding 123 we can use its left-to-right minima[1] to partition the remaining points into cells. Each cell must be decreasing, and the same is true for each row and each column, as noted by Claesson and Kitaev [25]. Therefore the permutation is uniquely determined by the number of points in each cell. If a cell is non-empty, then all the cells strictly above and strictly to the right of it will be empty. See, e.g., Figure 7.1 where we have five left-to-right minima and are assuming that A contains at least one point. This

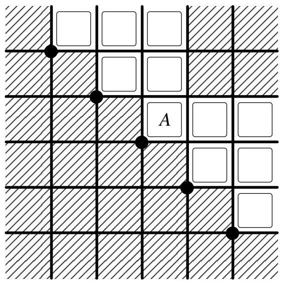


Figure 7.1: An avoider of 123 with five left-to-right minima where A contains at least one point

property allows us to construct a larger avoider from two smaller ones. See Figure 7.2 where F' has one more point than F. If we are adding the empty permutation, on the left, we instead add a left-to-right minimum. See Figure 7.3. This construction is reversible. Therefore, if

[1]A point (i, π_i) of a permutation π is called a *left-to-right minimum* if $\pi_j > \pi_i$ for all j in $\{1, 2, \ldots, i-1\}$.

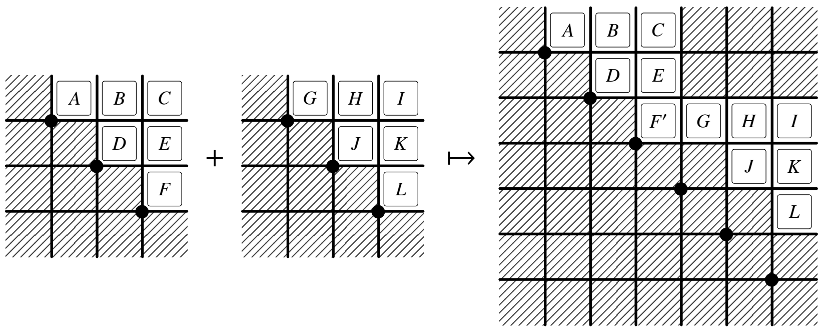


Figure 7.2: The sum of two 123-avoiding permutations

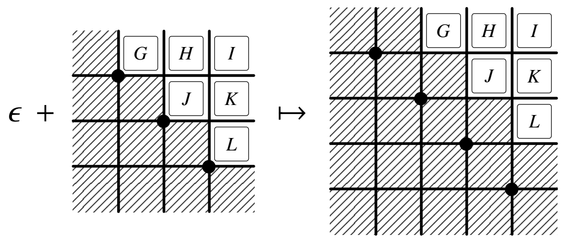


Figure 7.3: The sum of the empty permutation and a 123-avoiding permutation

we let $A(x)$ be the generating function for $\mathrm{Av}(123)$ then it is clear that it will satisfy

$$A(x) = 1 + x(A(x) - 1)A(x) + xA(x) = 1 + xA(x)^2,$$

which solves to give the generating function of the Catalan numbers.

The key to this proof of the enumeration of $\mathrm{Av}(123)$ is to consider the set of left-to-right minima, which in Bean, Claesson, and Ulfarsson [15] we called the *boundary* of the permutation. We also considered other types of boundaries that use left-to-right maxima, right-to-left minima or right-to-left maxima and even unions of these. In Bean, Tannock, and Ulfarsson [18], we extended the boundary idea further to enumerate other permutation classes and some subsets of permutations avoiding 1324.

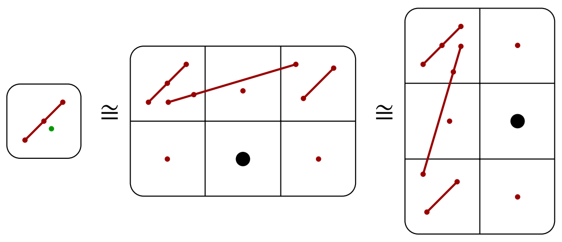
In the case of $\mathrm{Av}(1324)$, we considered the boundary formed by the left-to-right-minima and right-to-left maxima. As with the permutations above, given a permutation avoiding 1324 if we partition the remaining points not on the left-to-right-minima and right-to-left maxima boundary into cells, then each cell, row, and column must be decreasing. Every boundary for a permutation avoiding 1324 is a permutation avoiding 123. Given any boundary, i.e., any permutation avoiding 123, it is possible to enumerate the permutations in $\mathrm{Av}(1324)$ with that boundary. It corresponds to a tiling that is locally factorable, so is enumerated by the Tilescope algorithm. However, for this to lead to the enumeration of $\mathrm{Av}(1324)$ we would somehow need to consider all boundaries simultaneously as we did in our proof for $\mathrm{Av}(123)$.

Working with general boundaries seems difficult to automate as often there is no bound on how many points a boundary contains. If we were to find a way to bridge this gap and teach a computer to argue about a general boundary with an arbitrary number of points, it would be possible to recreate the argument above for enumerating $\mathrm{Av}(123)$ and hopefully many other permutation classes.

7.1.2 Interleaving factors

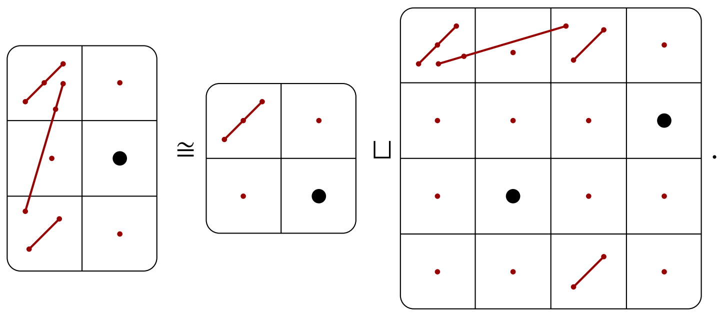
In Bean, Gudmundsson, and Ulfarsson [16, Proposition 14], we discussed a method for finding the structure of permutations in Av(123) that we hoped could be found by a future algorithm. That future algorithm we were discussing was the Tilescope algorithm, and it is able to rediscover the structure in the proposition. However, this structure is not a combinatorial specification because it uses a non-admissible constructor. We will present a slight variation, which leads to enumeration by adding a catalytic variable.

Applying the PointPlacement strategy to non-empty permutations avoiding 123 gives

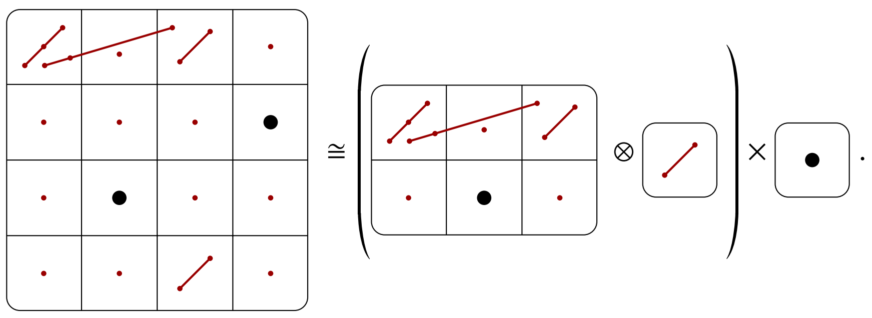

(7.1)

by placing either the rightmost or bottommost point. Let $F(x, y)$ be the generating function where the coefficient of $x^n y^k$ counts the number of length n permutations avoiding 123 with k points to the right of the minimum. From Equation (7.1), as the two tilings with placed points are the same up to inverse symmetry, this coefficient also counts the number of length n permutations avoiding 123 with k points below the rightmost point.

A permutation in Av(123) either has no points below the rightmost point or at least one which we can apply PointPlacement to, to give the combinatorial rule


(7.2)

If we were to apply Factors to the rightmost tiling in Equation 7.2, then only the two points would be removed. However, there are no obstructions across the cells $(2, 0)$ and $(2, 3)$ and if we let $\otimes$ be the operator that counts the interleavings of points in the cells, then


(7.3)

Equations (7.1), (7.2), and (7.3) imply that $F(x, y)$ satisfies the functional equation

$$F(x,y) = 1 + xF(x,1) + \frac{xy\left(F\left(x,\frac{1}{1-xy}\right)-1\right)}{1-xy}, \tag{7.4}$$

where the last term comes from counting the interleavings in Equation (7.3). Given the functional equation in Equation (7.4) it is a non-trivial exercise to show that $F(x, 1)$ is indeed the generating function for the Catalan numbers.

If we alter the Factors strategy to allow for such interleavings then structurally the combinatorial rule makes sense as given a gridded permutation from each factor the rule provides a recipe to build a set of gridded permutations from the original tiling. However, the enumerative information is harder to capture as the $\otimes$ constructor is no longer easy to understand. In some special cases, we can introduce catalytic variables to track the number of points in specific regions of tilings and write down a useful functional equation.

In the case of PointPlacement, in the forward direction, a single cell expands into multiple cells. If this equivalence is reversed, then a set of cells corresponds to a single cell. If the constructor interleaves one of these cells, then the information is now lost. If we only allow the forward direction of this, we can write down functional equations with similar substitutions as we did for $\mathrm{Av}(123)$. Doing this will give a polynomial time algorithm for the enumeration. However, the equations are hard to solve for a generating function for a permutation class.

By allowing interleavings Tilescope succeeded for all but 43 of the permutation classes with length four patterns that have not been enumerated by other methods in this thesis. The method for turning the Tilescope output with interleavings into a polynomial time algorithm has not been implemented.

7.2 Other combinatorial classes

The primary focus of this thesis has been automating the combinatorial exploration of permutation classes. This automation used the CombSpecSearcher algorithm, introduced in Chapter 3. CombSpecSearcher is a general purpose algorithm for performing combinatorial exploration on any combinatorial class.

In particular, it would be interesting to develop the appropriate strategies to automate the discovery of the results from Section 2.2 on bivincular patterns. There have already been some automatic methods for enumerating sets avoiding vincular patterns, Baxter and Pudwell [14]. The theory of gridded permutations in Chapter 4 should be extendable to bivincular patterns. It also seems possible to go one step further and define mesh gridded patterns akin to the definition of the mesh patterns defined by Brändén and Claesson [22]. The added difficulty with these types of patterns is that they lose the downwards closure property that classical permutation patterns have.

Outside of the field of permutation patterns, there are many other combinatorial classes for which combinatorial exploration could be automated. For example, as a BSc project that I supervised, Helgason and Robb [32] applied the CombSpecSearcher algorithm to two different combinatorial classes. The first was sets of binary string that avoid consecutive substrings. The second was pattern avoiding set partitions, using the definition given by Jelínek, Mansour, and Shattuck [35]. This is preliminary work but shows that the CombSpecSearcher is ready to be applied to other combinatorial classes.

7.3 Open questions

This section acts as a glossary for conjectures and questions asked throughout this thesis.

Every polynomial permutation class has a Struct cover by Theorem 2.4.1. Therefore, it follows that the Struct algorithm will theoretically terminate. However, the size of the peg permutations is not known and therefore whether running the Struct algorithm will terminate in a reasonable amount of time.

Question 7.3.1. Every polynomial permutation class is a disjoint union of peg permutations with filling vectors by Theorem 2.4.2. Is there a bound on the size of the peg permutations needed in this theorem?

A bound implies there is a bound on the size of Struct rules required in a Struct cover for a polynomial permutation class.

In Chapter 4, the theory of gridded permutations and tilings were developed. In Section 4.2, we found two tilings whose obstructions were minimal with respect to containment but defined the same set of gridded permutations.

Question 7.3.2. Does there exist a meaningful canonical form for a tiling? Moreover, if it exists, how can it be computed?

Lemma 4.1.1 partially answers this question as it provides a possible canonical form for the set of requirements. However, it is not clear what the canonical form for the set of obstructions is if it exists.

In Chapter 5 we used the CombSpecSearcher, introduced in Chapter 3, to find the insertion encoding and enumerate permutation classes that have a regular insertion encoding. Theorem 5.1.1 shows it can be seen from the basis of a permutation class as to whether or not it has a regular insertion encoding. In Section 5.2 we extended the algorithm to the separation insertion encoding that allows us to say which slots in a configuration must be inserted into first. In Section 5.3, we added another generalization, the top and bottom insertion encoding that allowed for inserting from both the top and bottom in a configuration.

Question 7.3.3. Does there exist a condition such as the one in Theorem 5.1.1 that characterizes the bases that will terminate with the separation insertion encoding? What about the top and bottom insertion encoding? If there is an answer to both of these questions, what about the separation top and bottom insertion encoding?

In Section 6.5 we introduced the notion of an elementary permutation class. Theorem 6.5.2 shows that this definition is same as permutation class being a disjoint union of valid generalized peg permutations. If follows that all polynomial permutation classes are elementary, Corollary 6.5.3.

Question 7.3.4. Which permutation classes are elementary? It is clear that an elementary permutation class' generating function will satisfy an algebraic system of equations with respect to the generating function of some its subclasses, but does it follow that the elementary permutation class has an algebraic generating function?

In Section 6.7 we defined a combinatorial system to be any set of combinatorial rules and showed how Tilescope can find a combinatorial system that can be used to enumerate $\text{Av}(1243, 1234, 1324, 1423, 2134, 2314)$, but Tilescope could not find a combinatorial specification.

Question 7.3.5. When can a combinatorial system can be used to enumerate a permutation class? Each combinatorial rule corresponds to an equation on the level of generating functions so that the question is the same as, when does a certain variable have a solution in a system of equations?

It would be more interesting if something can be said combinatorially. With this, it may make searching for combinatorial systems a less arduous task.

Bibliography

[1] M. H. Albert and M. D. Atkinson, "Simple permutations and pattern restricted permutations", *Discrete Math.*, vol. 300, no. 1-3, pp. 1–15, 2005.

[2] M. H. Albert, M. D. Atkinson, and R. Brignall, "Permutation classes of polynomial growth", *Ann. Comb.*, vol. 11, no. 3-4, pp. 249–264, 2007.

[3] M. H. Albert, M. D. Atkinson, and V. Vatter, "Counting 1324, 4231-avoiding permutations", *Electron. J. Combin.*, vol. 16, no. 1, Research Paper 136, 9, 2009.

[4] ——, "Inflations of geometric grid classes: three case studies", *Australas. J. Combin.*, vol. 58, pp. 24–47, 2014.

[5] M. H. Albert and R. Brignall, "Enumerating indices of Schubert varieties defined by inclusions", *J. Combin. Theory Ser. A*, vol. 123, pp. 154–168, 2014.

[6] M. H. Albert, S. Linton, and N. Ruškuc, "The insertion encoding of permutations", *Electron. J. Combin.*, vol. 12, Research Paper 47, 31, 2005.

[7] M. H. Albert, M. D. Atkinson, M. Bouvel, N. Ruškuc, and V. Vatter, "Geometric grid classes of permutations", *Trans. Amer. Math. Soc.*, vol. 365, no. 11, pp. 5859–5881, 2013.

[8] A. Arnarson, C. Bean, A. Bjarnason, U. Erlendsson, H. Ulfarsson, and S. Viktorsson, *PermPAL*, `http://permpal.ru.is/`, 2017.

[9] A. Arnarson, A. Bjarnason, U. Erlendsson, and S. Viktorsson, "PermPAL - Permutation Pattern Avoidance Library", *Skemman*, 2017. [Online]. Available: `http://hdl.handle.net/1946/28794`.

[10] E. Babson and E. Steingrímsson, "Generalized permutation patterns and a classification of the Mahonian statistics", *Sém. Lothar. Combin.*, vol. 44, Art. B44b, 18 pp. (electronic), 2000.

[11] F. Bassino, M. Bouvel, A. Pierrot, C. Pivoteau, and D. Rossin, "An algorithm computing combinatorial specifications of permutation classes", *Discrete Appl. Math.*, vol. 224, pp. 16–44, 2017.

[12] F. Bassino, M. Bouvel, A. Pierrot, and D. Rossin, "An algorithm for deciding the finiteness of the number of simple permutations in permutation classes", *Adv. in Appl. Math.*, vol. 64, pp. 124–200, 2015.

[13] ——, "Deciding the finiteness of the number of simple permutations contained in a wreath-closed class is polynomial", *Pure Math. Appl. (PU.M.A.)*, vol. 21, no. 2, pp. 119–135, 2010.

[14] A. M. Baxter and L. K. Pudwell, "Enumeration schemes for vincular patterns", *Discrete Math.*, vol. 312, no. 10, pp. 1699–1712, 2012.

[15] C. Bean, A. Claesson, and H. Ulfarsson, "Enumerations of permutations simultaneously avoiding a vincular and a covincular pattern of length 3", *J. Integer Seq.*, vol. 20, no. 7, Art. 17.7.6, 25, 2017.

[16] C. Bean, B. Gudmundsson, and H. Ulfarsson, "Automatic discovery of structural rules of permutation classes", 2017. eprint: `arXiv:1705.04109`.

[17] ——, *PermStruct*, `https://github.com/PermutaTriangle/PermStruct`, 2017.

[18] C. Bean, M. Tannock, and H. Ulfarsson, "Pattern avoiding permutations and independent sets in graphs", 2015. eprint: `arXiv:1512.08155`.

[19] A. Biere, *Lingeling, Plingeling and Treengeling Entering the SAT Competition*, 2013.

[20] M. Bóna, "The permutation classes equinumerous to the smooth class", *Electron. J. Combin.*, vol. 5, Research Paper 31, 12, 1998.

[21] M. Bousquet-Mélou, A. Claesson, M. Dukes, and S. Kitaev, "(2 + 2)-free posets, ascent sequences and pattern avoiding permutations", *J. Combin. Theory Ser. A*, vol. 117, no. 7, pp. 884–909, 2010.

[22] P. Brändén and A. Claesson, "Mesh patterns and the expansion of permutation statistics as sums of permutation patterns", *Electron. J. Combin.*, vol. 18, no. 2, Paper 5, 14, 2011.

[23] R. Brignall and J. Sliačan, "Juxtaposing Catalan permutation classes with monotone ones", *Electron. J. Combin.*, vol. 24, no. 2, Paper 2.11, 16, 2017.

[24] F. R. K. Chung, R. L. Graham, V. E. Hoggatt Jr., and M. Kleiman, "The number of Baxter permutations", *J. Combin. Theory Ser. A*, vol. 24, no. 3, pp. 382–394, 1978.

[25] A. Claesson and S. Kitaev, "Classification of bijections between 321- and 132-avoiding permutations", in *20th Annual International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2008)*, ser. Discrete Math. Theor. Comput. Sci. Proc., AJ, Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2008, pp. 495–506.

[26] A. Claesson and T. Mansour, "Enumerating permutations avoiding a pair of Babson-Steingrímsson patterns", *Ars Combin.*, vol. 77, pp. 17–31, 2005.

[27] A. Claesson, "Counting segmented permutations using bicoloured Dyck paths", *Electron. J. Combin.*, vol. 12, Research Paper 39, 18, 2005.

[28] R. Donaghey and L. W. Shapiro, "Motzkin numbers", *J. Combinatorial Theory Ser. A*, vol. 23, no. 3, pp. 291–301, 1977.

[29] S. Elizalde and T. Mansour, "Restricted Motzkin permutations, Motzkin paths, continued fractions and Chebyshev polynomials", *Discrete Math.*, vol. 305, no. 1-3, pp. 170–189, 2005.

[30] P. Flajolet and R. Sedgewick, *Analytic combinatorics*. Cambridge University press, 2009.

[31] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2016. [Online]. Available: `http://www.gurobi.com`.

[32] S. Helgason and J. Robb, *Identifying Combinatorial Structures for Binary Strings and Set Partitions*, 2018.

[33] C. Homberger and V. Vatter, "On the effective and automatic enumeration of polynomial permutation classes", *J. Symbolic Comput.*, vol. 76, pp. 84–96, 2016.

[34] S. Huczynska and V. Vatter, "Grid classes and the Fibonacci dichotomy for restricted permutations", *Electron. J. Combin.*, vol. 13, no. 1, Research Paper 54, 14 pp. (electronic), 2006.

[35] V. Jelínek, T. Mansour, and M. Shattuck, "On multiple pattern avoiding set partitions", *Adv. in Appl. Math.*, vol. 50, no. 2, pp. 292–326, 2013.

[36] T. Kaiser and M. Klazar, "On growth rates of closed permutation classes", *Electron. J. Combin.*, vol. 9, no. 2, Research paper 10, 20, 2002/03, Permutation patterns (Otago, 2003).

[37] S. Kitaev, "Multi-avoidance of generalised patterns", *Discrete Math.*, vol. 260, no. 1-3, pp. 89–100, 2003.

[38] D. Knuth, *The art of computer programming. Vol. 3*. Addison-Wesley, Reading, MA, 1998, pp. xiv+780, Sorting and searching, Second edition.

[39] D. Kremer, "Permutations with forbidden subsequences and a generalized Schröder number", *Discrete Math.*, vol. 218, no. 1-3, pp. 121–130, 2000.

[40] ——, "Postscript: Permutations with forbidden subsequences and a generalized Schröder number [Discrete Math. 218 (2000), no. 1-3, 121–130", *Discrete Math.*, vol. 270, no. 1-3, pp. 333–334, 2003.

[41] D. Kremer and W. C. Shiu, "Finite transition matrices for permutations avoiding pairs of length four patterns", *Discrete Math.*, vol. 268, no. 1-3, pp. 171–183, 2003.

[42] I. Le, "Wilf classes of pairs of permutations of length 4", *Electron. J. Combin.*, vol. 12, Research Paper 25, 26, 2005.

[43] P. A. MacMahon, "Combinatory Analysis", *Cambridge University Press, London*, 1915/16.

[44] T. K. Magnússon, "Forced permutation patterns and applications to coincidence classification of mesh patterns and enumeration of permutation classes", *Skemman*, 2018. [Online]. Available: `http://hdl.handle.net/1946/29902`.

[45] T. Mansour and M. Schork, "Wilf classification of subsets of eight and nine four-letter patterns", *J. Comb. Number Theory*, vol. 8, no. 3, pp. 257–283, 2016.

[46] ——, "Wilf classification of subsets of four letter patterns", *J. Comb. Number Theory*, vol. 8, no. 1, pp. 1–129, 2016.

[47] ——, "Wilf classification of subsets of six and seven four-letter patterns", vol. 9, Dec. 2017.

[48] T. Mansour and M. Shattuck, "Nine classes of permutations enumerated by binomial transform of Fine's sequence", *Discrete Appl. Math.*, vol. 226, pp. 94–105, 2017.

[49] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "SymPy: symbolic computing in Python", *PeerJ Computer Science*, vol. 3, e103, 2017.

[50] J. Pantone, "The enumeration of permutations avoiding 3124 and 4312", *Ann. Comb.*, vol. 21, no. 2, pp. 293–315, 2017.

[51] B. Salvy and P. Zimmermann, "GFUN: a maple package for the manipulation of generating and holonomic functions in one variable.", *INRIA*, 1992.

[52] N. Sloane, *The On-Line Encyclopedia of Integer Sequences*, published electronically at http://oeis.org, 2010.

[53] B. E. Tenner, "Repetition in reduced decompositions", *Adv. in Appl. Math.*, vol. 49, no. 1, pp. 1–14, 2012.

[54] V. Vatter, "Enumeration schemes for restricted permutations", *Combin. Probab. Comput.*, vol. 17, no. 1, pp. 137–159, 2008.

[55] ——, "Finding regular insertion encodings for permutation classes", *J. Symbolic Comput.*, vol. 47, no. 3, pp. 259–265, 2012.

[56] ——, "Small permutation classes", *Proc. Lond. Math. Soc. (3)*, vol. 103, no. 5, pp. 879–921, 2011.

[57] J. West, "Generating trees and forbidden subsequences", in *Proceedings of the 6th Conference on Formal Power Series and Algebraic Combinatorics (New Brunswick, NJ, 1994)*, vol. 157, 1996, pp. 363–374.

[58] H. S. Wilf, *generatingfunctionology*, Third. A K Peters, Ltd., Wellesley, MA, 2006, pp. x+245.

[59] ——, "What is an answer?", *Amer. Math. Monthly*, vol. 89, no. 5, pp. 289–292, 1982.

[60] D. Zeilberger, "Enumeration schemes and, more importantly, their automatic generation", *Ann. Comb.*, vol. 2, no. 2, pp. 185–195, 1998.