# PACKING AND COUNTING PERMUTATIONS

**Jakub Sliačan**

a thesis submitted to The Open University
for the degree of Doctor of Philosophy in Mathematics


February 2017

# Abstract

A permutation class is a set of permutations closed under taking subpermutations. We study two aspects of permutation classes. Enumeration and packing.

Our work on enumeration consists of two campaigns. First, we enumerate all juxtaposition classes of the form "Av($abc$) next to Av($xy$)", where $abc$ is a permutation of length three and $xy$ is a permutation of length two. We chose to represent elements from such a juxtaposition class by Dyck paths decorated with sequences of points. Context free grammars are then used to enumerate these decorated Dyck paths. Second, we classify the generating functions of $1 \times m$ permutation grid classes, where one cell is context-free and the remaining cells are monotone, as algebraic. We rely on properties of combinatorial specifications of context-free classes and use operators to capture juxtapositions. Repeated application of operators resolves $1 \times m$ grid classes for $m > 2$. We provide numberous examples and corollaries to both re-prove known results and yield new ones. Our methods are algorithmic and as such can be implemented on a PC.

In our work on packing, we consolidate what is currently known about packing densities of 4-point permutations and in the process improve the lower bounds for the packing densities of 1324 and 1342. We also provide rigorous upper bounds for the packing densities of 1324, 1342, and 2413. All our bounds are within $10^{-4}$ of the true packing densities. Together with the known bounds, this gives us a fairly complete picture of all 4-point packing densities. We also list a number of lower bounds and upper bounds for small permutations of length at least five. Our main tool for the upper bounds is the framework of flag algebras introduced by Razborov in 2007. We also present Permpack — a flag algebra package for permutations.

# Acknowledgements

I would like to thank my supervisor Robert Brignall for finding nice balance between guidance and freedom. I especially valued the opportunity to pursue side projects with little or no relationship to my thesis without the pressure of producing measurable results. It made mathematics more enjoyable. Thank you.

My acknowledgements go to the Department of Mathematics and Statistics at The Open University for funding my PhD and providing support in various forms throughout the past three years.

Special thanks go to my office mates who tolerated my smelly running clothes as well as my constant chewing in our office: Grahame Erskine, Michael Ewetola, Jay Fraser, and Olivia Jeans. I am grateful to David Bevan for advice and welcome when I first came to the department and for maths conversations. Thanks also go to Lax Chan, Argyris Christodoulou, Ioannis Dourekas, Vasso Evdoridou, Matthew Jacques, Robert Lewis, Alison Maidment, David Marchant, Maha Moustafa, David Martí-Pete, Tony Royle, Margaret Stanier, Brigitte Stenhouse, and James Tuite.

Additionally, I would like to thank my co-authors, collaborators, and influences outside of my department: Dan Kráľ, Oleg Pikhurko, Jozef Skokan, Kostas Tyros, Walter Stromquist, and Michael Albert.

Thanks also go to my parents and Veronika for being supportive throughout.

Lastly, I thank Fiona who helps me stay mathematically curious.

# Declarations

There are five chapters in this thesis.

1. Chapter 1 consists of a general exposition of the area of permutation patterns and the two viewpoints of the area that we take in this thesis: enumeration and packing. Most of the work in this chapter follows expositions in other theses such as Bevan's [Bev15b] TODO: add more references.

2. Chapter 2 consists of joint work with Robert Brignall. The corresponding paper [BS17] is published in *Electronic Journal of Combinatorics*. We use PermLab [Alb12], Mathematica [Incb] and Sage [Dev17] for computations.

3. Chapter 3 consists of joint work with Robert Brignall. We use PermLab [Alb12] and Mathematica [Incb] for our computations.

4. Chapter 4 consists of joint work with Walter Stromquist. The corresponding paper is published in *Discrete Mathematics and Theoretical Computer Science* in *Permutation Patterns 2016* special issue. We make extensive use of software: Flagmatic package [Vau13], Mathematica [Incb], Sage [Dev17], and our own Permpack [Sli16].

5. Chapter 5 consists of description of Permpack that the author wrote for the work in Chapter 4. Permpack was written to resemble Flagmatic in order to make it easier to use alongside Flagmatic. However, no Flagmatic code was used, nor were any algorithms taken into Permpack. Permpack is a Sage [Dev17] package.

None of the results appear in any other thesis and all co-authors have agreed with inclusion of joint work in this thesis.

# Contents

# Chapter 1

# General Introduction

*Enumerating permutations is sometimes hard and usually tedious. Packing permutations is often hard and always tedious.*

— folklore

This entire thesis is concerned with only one kind of object — *permutation*. We treat permutations as patterns or words that use every letter in the alphabet exactly once. The alphabet being $[n] := \{1, \ldots, n\}$. In fact, we study permutation classes rather than permutations themselves. These are collections of permutations closed under taking subpermutations. There are several natural approaches to studying permutation classes. Let $\mathcal{C}$ be a permutation class. Then one can enquire about the properties of a what a typical object from $\mathcal{C}$ "looks" like? Alternatively, one could be interested in how many permutations of each length are there in $\mathcal{C}$? Yet another different approach would be to ask questions such as what is the maximum number of inversions that a permutation in $\mathcal{C}$ can have? While all three are interesting directions of study, we focus on questions of the second and third kinds only.

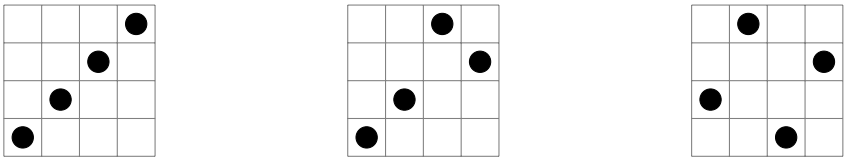
The enumerative approach to permutation classes has been quite dominant in the permutation patterns community. There are several works that survey this area chronologically and systematically. We point to the chapter *Permutation Classes* by Vatter [V.15] in the Handbook of Enumerative Combinatorics. For further book material, refer to the references therein. On the other hand, additional

surveys of the field can be found in the conference proceedings of Permutation Patterns 2007 [LRV10]. The contributions relevant to this thesis are *A survey of simple permutations* by Brignall [Bri10], *An introduction to structural methods in permutation patterns* by Albert [Alb10], and parts of *Some general results in combinatorial enumeration* by Klazar [M.10]. Another relevant survey is *Some open problems on permutation patterns* by Steingrimsson [Ste12]. The general background of enumerative combinatorics from the perspective of generating functions via the *symbolic method* is best treated in *Analytic Combinatorics* by Flajolet and Sedgewick [FS09].

Permutation packing has been less prevalent among research topics in the area of permutation patterns. The single best survey article, containing new (at that time) results, is *On packing densities of permutations* by Albert, Atkinson, Handley, Holton, and Stromquist [AAH+02]. Although there have been significant advances in permutation packing area since 2002, there have not been many of them. Hence the article is still relevant in 2018.

## 1.1 Concepts and definitions

We now proceed to define key concepts needed throughout the thesis. We postpone the particular definitions needed in individual chapters to those chapters. A *pattern* of length $k$, where $k \leq n$, is a $k$-tuple of distinct integers from $[n] := \{1, \ldots, n\}$. A pattern of length $n$ is called a *permutation*. We write tuples as strings: 1324 stands for $(1, 3, 2, 4)$. Two patterns $\pi$ and $\sigma$ of length $k$ are *identical*, if $\pi[i] = \sigma[i]$ for all $i \in [k]$. They are *order-isomorphic* if for all pairs of indices $i, j$, it holds that $\pi[i] < \pi[j]$ if and only if $\sigma[i] < \sigma[j]$. For a set $I = \{i_1, \ldots, i_m\}$ of $m$ indices from $[n]$, the *sub-pattern* $\pi[I]$ is the $m$-tuple $\pi[i_1]\pi[i_2]\cdots\pi[i_m]$. By overloading the notation slightly, we also use $\pi[I]$ to refer to the *subpermutation* of length $m$ which is order-isomorphic to the sub-pattern $\pi[I]$. Finally, we do not distinguish between different representations of the same permutation. For example, 2413 and its plot on the grid in Figure 1.1 will be referred to as 2413 interchangeably. Let $\mathcal{F}$ be a set of *forbidden* permutations. We say that permutation $\pi$ is $\mathcal{F}$*-free* if no $\phi \in \mathcal{F}$ is a subpermutation of $\pi$. Such $\pi$ is also said to *avoid* $\mathcal{F}$ or be *admissible*.



(a) 1234 (b) 1243 (c) 2413

Figure 1.1: Pictorial representations of selected permutations.

### 1.1.1 Special permutations

An *interval* refers to a contiguous set of integers, e.g. 3645 is an interval in 213645. A permutation $\pi$ is *simple* if it does not contain any non-trivial intervals. For instance, 2413 is simple while 1243 is not (both 12 and 43 are intervals). We call $\pi$ an *inflation* of $\sigma$ if it can be obtained from $\sigma$ by substituting points of $\sigma$ for permutations. We denote $\pi$ as inflation of $\sigma$ by $\pi = \sigma[\alpha_1, \ldots, \alpha_{|\sigma|}]$, where $\alpha_1, \ldots, \alpha_{|\sigma|}$ are permutations that inflate $\sigma$ into $\pi$. Consider the example of 1243 as an inflation of 12, so $1243 = 12[12, 21]$. In this case, it is also an inflation of 12 by 1 and 132 as in $1243 = 12[1, 132]$. There is a fundamental result by Albert and Atkinson [AA05] which says that if $\sigma$ is of length at least three and simple, then for any $\pi$ which is an inflation of $\sigma$ there is always a unique way to inflate $\sigma$ into $\pi$. Hence, 12 and 21 are special.

A *decreasing (increasing) permutation* of length $k$ is $k \ldots 321$, respectively $123 \ldots k$. A permutation $\pi$ is *layered*, if it is an increasing sequence of decreasing permutations. To be exact, a layered permutation $\pi$ is a concatenation of smaller permutations $\pi = \pi_1 \pi_2 \ldots \pi_\ell$ such that for all $1 \leq i \leq \ell$, $\pi_i$ is a decreasing sequence of consecutive integers satisfying the following: if $x \in \pi_i$ and $y \in \pi_j$ with $i < j$, then $x < y$. For instance, 321465987 can be partitioned as 321|4|65|987, so it is layered. On the other hand, 2413 is not layered. This brings us to the notion of sum and skew-sum of permutations. Let $\pi_1$ and $\pi_2$ be permutations of lengths $k$ and $\ell$. We say that $\pi$ is a *sum of* $\pi_1$ *and* $\pi_2$, denoted by $\pi = \pi_1 \oplus \pi_2$, if $\pi$ consists of two intervals $\pi[1] \cdots \pi[k]$ and $\pi[k+1] \cdots \pi[\ell]$ such that $\pi[1] \cdots \pi[k]$ is order-isomorphic to $\pi_1$, $\pi[k+1] \cdots \pi[\ell]$ is order-isomorphic to $\pi_2$, and $\pi[i] < \pi[j]$ for all $i \leq k$ and $j > k$. Similarly, $\pi$ is a *skew-sum of* $\pi_1$ *and* $\pi_2$, denoted by $\pi = \pi_1 \ominus \pi_2$, if $\pi$ consists of the two intervals as above, except this time we require that $\pi[i] > \pi[j]$ for all $i \leq k$ and

$j > k$. A permutation is called *sum-indecomposable* if it cannot be expressed as a sum of two non-empty permutations. Analogously, *skew-indecomposable* permutations cannot be expressed as skew sums of non-empty permutations. With this new notation in place, a layered permuation with $k$ layers is $\pi = \pi_1 \oplus \cdots \oplus \pi_k$ such that all $\pi_i$ are decreasing permutations. A permutation is called *separable*, if it can be obtained from single points by repeated application of sum and skew-sum. For instance, $42315 = (1 \ominus (1 \oplus 1) \ominus 1) \oplus 1$ is separable, but 2413 is not.

### 1.1.2 Permutation classes

A *permutation class* $\mathcal{C}$ is a set of permutations which is closed under taking subpermutations, i.e. if $\pi$ is in $\mathcal{C}$ and $\sigma \subseteq \pi$, then $\sigma$ is also in $\mathcal{C}$. Given that the subpermutation relation is a partial order on a permutation class $\mathcal{C}$, there is a minimal set of forbidden permutations called the *basis* $\mathcal{B}$ of $\mathcal{C}$. We also write $\mathcal{C} = \mathrm{Av}(\mathcal{B})$ to make explicit the fact that $\mathcal{C}$ is the set of avoiders of $\mathcal{B}$. For example, the class $\mathrm{Av}(231, 312)$ is the class of layered permutations. Similarly, $\mathrm{Av}(2413, 3142)$ is the class of separable permutations. The subset of a class $\mathcal{C}$ containing only permutations of length $n$ is referred to by $\mathcal{C}_n$. Hence, $\mathcal{C} = \bigcup_{n\geq 0} \mathcal{C}_n$. We use $|\mathcal{C}_n|$ to denote the number of elements of length $n$ in $\mathcal{C}$.

To enumerate a permutation class $\mathcal{C}$ means to provide a sequence $(a_n)_{n\geq 0}$ such that $a_n = |\mathcal{C}_n|$. Given that $(a_n)_n$ has infinitely many terms, we need a clever data structure to store it in finite memory. A *generating function* $C(z)$ of $\mathcal{C}$ is a formal power series $C(z) = \sum_{n\geq 0} a_n z^n$ with the coefficient of $z^n$ being the $n$-th term of the sequence. Ideally we would know the closed form of $C(z)$, e.g. the closed form of $C(z) = \sum_{n\geq 0} z^n = 1/(1-z)$. A rational generating function is one whose closed form is a ratio of two polynomials. Algebraic generating function $f = f(z)$ is a root of a polynomial equation in $f$ and $z$. As a shortcut, when we say that a generating function enumerating $\mathcal{C}$ is rational/algebraic/etc., we mean that the closed form of the formal power series storing the counting sequence which enumerates $\mathcal{C}$ is rational/algebraic/etc.

It may not always be possible to find the generating function for a class, but a cruder way of "enumerating" a permutation class is by determining its growth rate. The *growth rate* of a class $\mathcal{C}$ is denoted by $\mathrm{gr}(\mathcal{C})$ and is defined as below

provided that the limit exists.

$$\mathrm{gr}\left(\mathcal{C}\right) = \lim_{n\to\infty} \sqrt[n]{|\mathcal{C}_n|}$$

Naturally, if the above limit does not exist despite being conjectured to always exist, we speak about an *upper growth rate* of $\mathcal{C}$ defined as $\limsup_{n\to\infty} \sqrt[n]{|\mathcal{C}_n|}$ and a *lower growth rate* of $\mathcal{C}$ defined as $\liminf_{n\to\infty} \sqrt[n]{|\mathcal{C}_n|}$. Conveniently, Marcus and Tardos [MT04] proved that the growth rate is well-defined for permutation classes with basis of size one — also called *principal* classes. Interestingly, there were conjectures about the growth rates of principal classes and all were refuted by this point in time. First Arratia [Arr99] conjectured that for every permutation $\sigma$, $\mathrm{gr}\left(\mathrm{Av}(\sigma)\right) \leq (k-1)^2$. However, in 2006 Albert, Elder, Rechnitzer, Westcott, and Zabrocki [AER+06] disproved this by showing that $\mathrm{gr}\left(\mathrm{Av}(1324)\right) > 9.47 > (4-1)^2$. Bóna first suggested in [Bó05] that layered permutations may be the easiest to avoid, i.e. their principal avoider classes tend to have highest growth rates, an impression strengthened by the result of Albert et al. [AER+06]. However, in 2013 Fox [Fox13] confirmed what we knew already: that we do not understand the growth rates of permutation classes very well. His result states that almost all patterns $\sigma$ of length $k$ have growth rates of a completely different order than we thought: $\mathrm{gr}\left(\mathrm{Av}(\sigma)\right) = 2^{\Omega(k)}$. Although this was tangential to the core topic of the thesis, it shows that permutation patterns form an exciting area to study.

# Part I

# Enumeration

This chapter contains results on enumeration of permutation grid classes. They are useful given that one of the approaches to enumerating permutation classes is through permutation *grid classes*. These are permutation classes themselves but offer additional insight into structure of the permutations in them. For instance, if a permutation $\sigma$ can be split by a vertical line into a left part and a right part so that the left part (as a subpermutation of $\sigma$) avoids 21 and the right part avoids 12, then $\sigma$ belongs to $\mathcal{C} = \mathrm{Av}(21|12)$, a $1 \times 2$ grid class with the left cell being $\mathrm{Av}(21)$ and the right cell being $\mathrm{Av}(12)$. All permutations in $\mathcal{C}$ can be split this way, and all permutations that admit such gridding belong to $\mathcal{C}$.

The grid class that we just described, in our notation $\mathrm{Av}(21)|\mathrm{Av}(12)$, was enumerated by Atkinson [Atk98]. In [Atk99], Atkinson used grid classes to enumerate other classes such as $\mathrm{Av}(132, 4321)$, $\mathrm{Av}(321, 2134)$, and $\mathrm{Av}(321, 1324)$ (Bevan calls these "skinny grid classes" in his thesis [Bev15b]). In recent years, grid classes were critical to enumeration of two-by-four classes with two basis elements of length four. See Pantone [Pan17] for enumeration of $\mathrm{Av}(3124, 4312)$. Albert, Atkinson, and Brignall used grid classes to enumerate $\mathrm{Av}(2143, 4231)$ in [AAB11] and three other two-by-four classes in [AAB12]. Albert, Atkinson, and Vatter [AAV14] use a special kind of grid classes to enumerate three specific permutation classes. Another paper making use of grid classes to enumerate $\mathrm{Av}(4231, 35142, 42513, 351624)$ is by Albert and Brignall [AB14]. We also mention Bevan's enumeration of $\mathrm{Av}(4213, 2143)$ in [Bev17] which utilizes permutation grid classes.

Recall that a growth rate of a permutation class $\mathcal{C}$ is defined as $\lim_n \sqrt[n]{|\mathcal{C}_n|} = \liminf_n \sqrt[n]{|\mathcal{C}_n|} = \limsup_n \sqrt[n]{|\mathcal{C}_n|}$ if it exists. Grid classes have been central to Vatter's proof [Vat11] of the fact that there are only countably many growth rates of permutation classes below $\xi \approx 2.30522$ while there are uncountably many growth rates arbitrarily close to $\xi$. Two follow-up papers of Vatter [Vat16], and Pantone and Vatter [PV16] make use of grid classes as well.

Apart from enumerating permutation classes, several other applications of grid classes exist, among them [AAB11] and [Bev17]. We give further examples, with accompanying commentaries, in the next paragraph. For a comprehensive introduction to grid classes and their further uses, see Bevan's PhD thesis [Bev15b],

Sections 2 and 6 in Part I, as well as parts of the general introduction in Section 1 of Part I.

Because of their more general applicability, the study of grid classes in their own right has emerged in a few directions. For instance, it is conjectured that all monotone grid classes are finitely based, but this is only known for a few special cases, most notably those whose row-column graph is acyclic [AAB+13], and a few other special cases (see [AB16, Atk99, Wat07, Bev15b]). In another direction, the role of grid classes with respect to partial well-ordering has been explored in e.g. [Bri12, MV03, VW11]. Finally, while the asymptotic enumeration of monotone grid classes was answered completely by Bevan [Bev15a], exact enumeration is harder, primarily due to the difficulty of handling multiple griddings: that is, enumerating 'griddable' objects rather than 'gridded' ones. One general result here is that all geometric grid classes have rational generating functions [AAB+13], but the move from 'gridded' to 'griddable' is nonconstructive, instead relying on properties of regular languages.

Since grid classes are often used on the way to enumerating other permutation classes, it would be convenient to be able to enumerate grid classes. Ideally, we would have exact enumerations of classes of the form shown in Figure 1.2 — the most generic form of grid classes. For the current level of discussion, Figure 1.2 also suffices as a definition of a grid class (a grid of permutation classes).

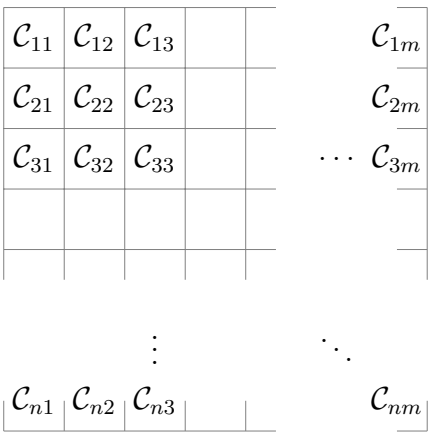
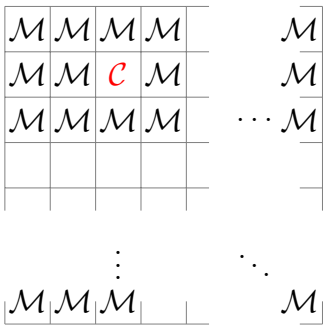
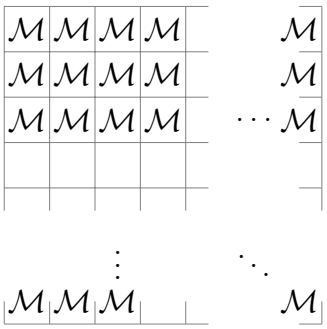



Figure 1.2: A generic format of a generalised grid class, where $\mathcal{C}_{ij}$ are arbitrary but fixed permutation classes.

The current state of affairs is much more grim. We cannot even enumerate grid classes of the form shown in Figure 1.3a or in Figure 1.3b. However, there are several important results in this direction. For instance, due to Bevan [Bev15a] we at least know the growth rates of monotone grid classes (where every cell is monotone, like Figure 1.3b). The growth rates are equal to the square of the spectral radius of a certain associated row-column graph.




(a) All cells are monotone except for $\mathcal{C}$, which is a more complex class.




(b) All cells are monotone (increasing or decreasing).

Figure 1.3: We cannot enumerate either of the two grid classes in 1.3a and 1.3b.

Approaching the topic from another angle, Albert, Atkinson, Bouvel, Ruškuc and Vatter [AAB+13] proved that geometric monotone grid classes are enumerated by *rational* generating functions. Notice that this result is different from the exact enumeration type of results. And that along two dimensions. First, the authors do exact enumeration in the usual constructive fashion. Second, they assume certain niceness of the grid class — the monotone classes are geometric and therefore the grid avoids cycles. Still, their result is important and goes to show how unreasonable it is at this point to ask for exact enumeration of arbitrary grid classes.

Lastly, Bevan [Bev15b] conjectured the generating functions of monotone increasing grid classes of dimension $1 \times k$ for some $k$. He also provides a method for enumerating $1 \times k$ monotone grid classes in the sense that for any fixed $1 \times k$ monotone grid class, he gives a finite procedure that enumerates it.

The aim of Part I of this thesis is to make progress on describing permutation grid classes along the lines of previous research. In Chapter 2 we pick the simplest

possible non-trivial grid classes of the form shown in Figure 1.3a and enumerate them exactly. They are $1 \times 2$ grid classes, also referred to as juxtapositions, of a Catalan class $\mathcal{C}$ with a a monotone class $\mathcal{M}$. In Chapter 3 we choose to prove a result similar in character to that of [AAB+13]. We show that all $1 \times m$ monotone grid classes with one cell substituted for a context-free class $\mathcal{C}$ admit algebraic generating functions. Our methods allow us to enumerate several new grid classes exactly.

# Chapter 2

# Simple juxtapositions

As mentioned earlier, in this chapter we enumerate all juxtaposition classes of the form "Av($abc$) next to Av($xy$)", where $abc$ is a permutation of length three and $xy$ is a permutation of length two. We use Dyck paths decorated by sequences of points to represent elements from such a juxtaposition class. Context free grammars are then used to enumerate these decorated Dyck paths.

## 2.1 Introduction

Juxtapositions are a simple special case of permutation grid classes. To keep this chapter as self-contained as possible, we define juxtapositions and Catalan classes in a particular way that is beneficial for the work in this chapter.

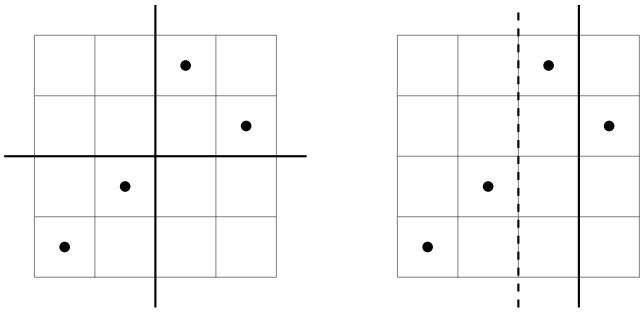


Figure 2.1: On the left is the unique gridding of 1243 by the gridding matrix $M = \begin{pmatrix} \emptyset & \text{Av}(12) \\ \text{Av}(21) & \emptyset \end{pmatrix}$. On the right are the two griddings of 1243 by $M = \begin{pmatrix} \text{Av}(21) & \text{Av}(12) \end{pmatrix}$.

Recall that each permutation in a *grid class* can be drawn into a grid so that the subpermutation in each box is in the class specified by the corresponding cell in a *gridding matrix*. See Figure 2.1 for an example of permutations from a *monotone grid class* (where each cell in the gridding matrix is a 21-avoider, 12-avoider, or empty).

As a first step towards enumerating more general grid classes, in this paper we replace one cell in the gridding matrix $M$ of a monotone grid class by a *Catalan class*, that is, one avoiding a single permutation of length 3. For simplicity, we restrict our attention to $1 \times 2$ grids, although the techniques presented here could be used in larger grids. These $1 \times 2$ grid classes are also referred to as *juxtapositions* — in our case a Catalan class in the left cell, and a monotone class in the right one.

$$\begin{aligned}
&\mathrm{Av}(213|21),\ \underline{\mathbf{Av(231|12)}} &&\overset{\theta}{\longleftrightarrow}&&\ \mathrm{Av}(123|21),\ \underline{\mathrm{Av}(321|12)}\\
&\mathrm{Av}(123|12),\ \underline{\mathbf{Av(321|21)}} &&\overset{\psi}{\longleftrightarrow}&&\ \mathrm{Av}(213|12),\ \underline{\mathrm{Av}(231|21)}\\
&\mathrm{Av}(132|12),\ \underline{\mathbf{Av(312|21)}} &&\overset{\phi}{\longleftrightarrow}&&\ \mathrm{Av}(132|21),\ \underline{\mathrm{Av}(312|12)}
\end{aligned}$$

Table 2.1: Each row contains equinumerous classes. Classes in pairs (separated by commas) are equinumerous by symmetry. Left column and right column (of the same row) are equinumerous by one of the bijections $\theta, \psi, \phi$, see Section 2.4. Each bijection describes a correspondence between the underlined classes in the given row. The classes in bold are enumerated via context-free grammars.

Let $P_i$ and $S_j$ be permutations, for all $1 \le i \le k$ and $1 \le j \le l$. Also, let $\mathbf{U} = \mathrm{Av}(P_1, \ldots, P_k)$ and $\mathbf{V} = \mathrm{Av}(S_1, \ldots, S_\ell)$ be classes of permutations that avoid $P_1, \ldots, P_k$ and $S_1, \ldots, S_\ell$, respectively. A *juxtaposition class* $\mathbf{W} = \mathrm{Av}(P_1, \ldots, P_k \mid S_1, \ldots, S_\ell)$ of classes $\mathbf{U}$ and $\mathbf{V}$ is the set of permutations whose form is $AB$ with $A \in \mathbf{U}$ and $B \in \mathbf{V}$. We say that $\mathbf{U}$ is on the left-hand side (LHS) of $\mathbf{W}$ and that $\mathbf{V}$ is on the right-hand side (RHS) of $\mathbf{W}$. The diagram on the right in Figure 2.1 shows a permutation 1342 from the juxtaposition class Av(21|12), together with two possible griddings of 1342. Equipped with the definition of a juxtaposition, we can now refer to Table 2.1. It schematizes the relationships between juxtapositions $\mathrm{Av}(P|S)$, where $P$ is of length three and $S$ of length two or vice versa. Out of all these, only twelve are essentially distinct (not symmetries of each other), and all of them have the Catalan class $\mathrm{Av}(abc)$ on the same side in the juxtaposition.

So, without loss of generality, we assume a Catalan class to be on the left and a monotone class on the right. By further symmetries, these twelve juxtaposition classes can be coupled into equinumerous pairs. See Table 2.1. To the best of our knowledge, only two of the twelve classes in Table 2.1 have been enumerated — Av(231|12) by Bevan [Bev17] (and hence Av(213|21)) and Av(321|12) by Miner [Min16] (and hence Av(123|21)). The goal of this paper is to enumerate the remaining two juxtaposition classes in bold in Table 2.1 and find bijections between the pairs of underlined classes. This completes the enumeration for all juxtapositions of a Catalan class with a monotone one.

In section 2.2 we introduce the concepts that we need and demonstrate them on an example. Section 2.3 contains enumerations of the boldface classes Av(231|12), Av(321|21) and Av(312|21). Bijections between the underlined classes are presented in Section 2.4. We mention open questions in Section 2.5.

## 2.2 Definitions and overview

We assume the reader is familiar with elementary definitions. They are conveniently presented in a note by Bevan [Bev15c]. We refer to the three juxtaposition classes in bold in Table 2.1 as $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, respectively. We list them below together with their representations in terms of bases, using [Atk99].

$$\begin{array}{lcll} \mathbf{A} & := & \mathrm{Av}(231|12) & = & \mathrm{Av}(2314, 2413, 3412) \\ \mathbf{B} & := & \mathrm{Av}(321|21) & = & \mathrm{Av}(4321, 32154, 42153, 52143, 43152, 53142) \\ \mathbf{C} & := & \mathrm{Av}(312|21) & = & \mathrm{Av}(4132, 4231, 31254, 41253) \end{array}$$

A quick enumeration (using PermLab [Alb12]) of the first twelve elements in $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ yields the following sequences.

$\mathbf{A}$: A033321 1, 2, 6, 21, 79, 311, 1265, 5275, 22431, 96900, 424068, 1876143
$\mathbf{B}$: A278301 1, 2, 6, 23, 98, 434, 1949, 8803, 39888, 181201, 825201, 3767757
$\mathbf{C}$: A165538 1, 2, 6, 22, 88, 367, 1568, 6810, 29943, 132958, 595227, 2683373

We observe that the sequence A165538 enumerates both classes $\mathbf{C}$ and Av(4312, 3142). The latter is studied as Example 2 in [AAV14]. We do not know of a straightforward reason for these to be equinumerous.

A *Dyck path* of length $2n$ is a path on the integer grid from $(0,0)$ to $(n,n)$ where each step is either $(k,m) \to (k+1,m)$ or $(k,m) \to (k,m+1)$. A Dyck path must stay on one side of the diagonal, i.e. it is not allowed to cross but can *touch* the diagonal. See Figure 2.2 for an example. The orientation of the Dyck path in our definition is arbitrary, and we will use the term "Dyck path" to refer to any of the four symmetries of a Dyck path (above/below the diagonal and top-to-bottom/bottom-to-top). In this paper, we only need two kinds of Dyck paths: those that use *up* and *right* steps (bottom-left to top-right Dyck paths that stay above the diagonal) and those that use *down* and *left* steps (top-right to bottom-left Dyck paths that stay below the diagonal). We will always specify which one of the two cases we work with.

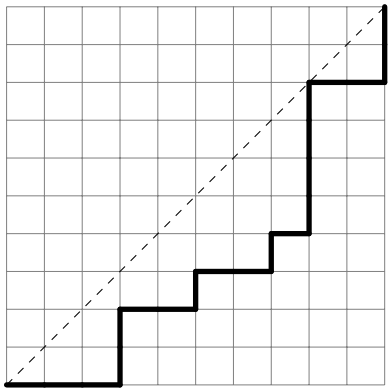


Figure 2.2: Example of a Dyck path of length 20.

Let $\mathcal{D}$ be Dyck path of length $2n$ from top right to the bottom left corner of the grid. Let it be below the diagonal. *Offsetting* a diagonal means considering a line with slope one from $(1,0)$ to $(n,n-1)$ on a square grid, instead of the original diagonal $(0,0)$ to $(n,n)$ (in case $\mathcal{D}$ is above the diagonal, we offset in the opposite direction). Figure 2.3 illustrates this with the dotted line moving downwards along the \-diagonal from 3rd to 4th and from 4th to 5th subfigure. An *excursion* in $\mathcal{D}$ is the part of the Dyck path $\mathcal{D}$ which meets the diagonal only at the beginning and at the end.

It is well known that permutations of length $n$ in Av(231) are in one-to-one correspondence with Dyck paths of length $2n$. And so are permutations of length

$n$ in Av(321). There are a number of bijections between these Catalan objects. We now fix one for each correspondence and describe them in detail.

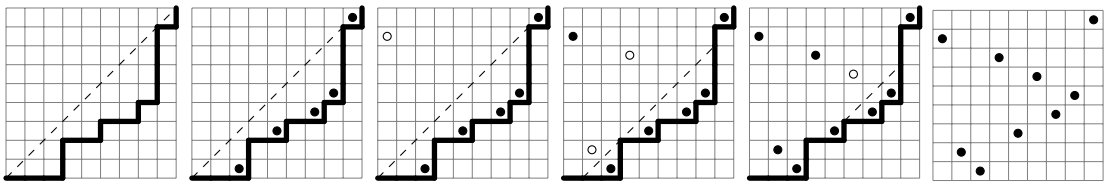


Figure 2.3: An example of a reversible process (left to right) of associating a unique 231-avoider with a given Dyck path. [ABRV16].

This paragraph is best read alongside Figure 2.3. To obtain a 231-avoider of length $n$ from a Dyck path $\mathcal{D}$, one places a point into each *corner* (a down step followed by a left step) to obtain the right-to-left minima of the permutation to be constructed. This is the second diagram from the left in Figure 2.3. We place the remaining points of the permutation for each excursion of $\mathcal{D}$ separately. Given such an excursion, the first and last steps are not consecutive/adjacent and must be a down step and a left step, respectively. Insert a point in the square where the respective row and column of these steps meet. This is documented in the third diagram of Figure 2.3. The points inserted in a given step are marked by a circle. Do the same for each excursion of $\mathcal{D}$. In the next step offset the diagonal, disregard the points above the new diagonal, and repeat the process. It is easy to check that this procedure is correct (gives a 231-avoider) and reversible, since the corners of the Dyck path are right-to-left minima.

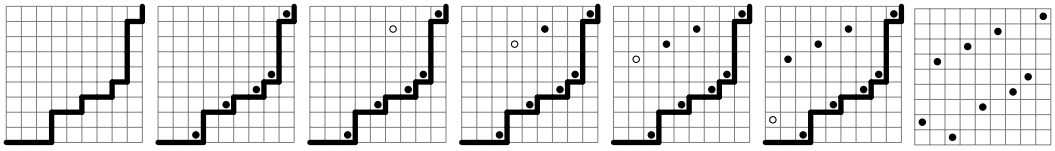


Figure 2.4: An example of a reversible process (left to right) of associating a unique 321-avoider with a given Dyck path.

This paragraph is accompanied by Figure 2.4. To obtain a 321-avoider of length $n$ from a Dyck path $\mathcal{D}$, place right-to-left minima as in the case above (231-avoider to Dyck path). Then, find the first down step that is not immediately succeeded

by a left step and find the first left step that is not immediately preceded by a down step. Place a point in the square where the row and column of these two meet. This step is shown in the third diagram in Figure 2.4 — the new point at each step is denoted by a circle. Add the rest of the points for the further down steps and left steps in the same fashion.

Having fixed bijections between Dyck paths and 231-avoiders and Dyck paths and 321-avoiders, we will refer to the Dyck path *corresponding* to a permutation $P$, and vice versa, whenever one of the bijections transforms one into the other.

A *context-free grammar (CFG)* is a formal grammar that describes a language consisting of only those words which can be obtained from a starting string by repeated use of permitted production rules/substitutions. Formally, a context-free grammar is a four-tuple $(V, \Sigma, R, \mathsf{S})$: a finite set of *variables* $V$, a set of terminal characters (symbols) $\Sigma$, a finite relation $R$ from $V$ to $(V \cup \Sigma)^*$ where $*$ is the Kleene star, and the start variable $\mathsf{S}$, $\mathsf{S} \in V$. The pairs in $R$ are called *production rules*, because they specify what we are allowed to substitute into a given variable. In our context-free grammars, the starting symbol is always $\mathsf{S}$ (unless CFG describes a Catalan object, then it is $\mathsf{C}$) and $\epsilon$ denotes an empty word. A good reference for formal languages is [HMU01].

**Example 2.2.1** (Enumerating Av(231))**.** Enumerating this class serves as a template for how we intend to use CFGs in Section 2.3. Recall that Dyck paths are in one-to-one correspondence with 231-avoiding permutations. We use this fact to enumerate permutations in Av(231) by counting Dyck paths. A Dyck path corresponding to a 231-avoider (we start in the bottom left corner, and stay below the diagonal) is either empty or starts with a step right, followed by an arbitrary Dyck path, followed by a step up, followed by an arbitrary Dyck path. Denote by $\mathsf{C}$ a Dyck path ($\mathsf{C}$ stands for Catalan object), by $\epsilon$ the empty Dyck path, and by $\mathsf{U}$, $\mathsf{R}$ up step and right step respectively. Then the CFG for Dyck paths (and hence also for 231- and 321-avoiders) looks as follows

$$\mathsf{C} \to \epsilon \mid \mathsf{RCUC}.$$

As written, this grammar is unambiguous, and can be transcribed syntactically into a functional equation. We let $z$ count the number of *up* steps (or alternatively

the number of right steps) in the Dyck path. We find that $c = c(z)$ satisfies the relation below

$$c = 1 + zc^2.$$

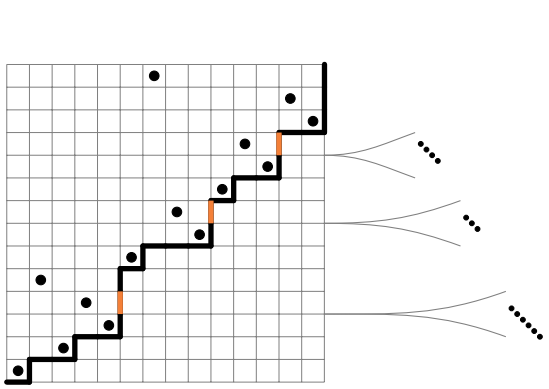
Consequently, we obtain $c$ as a function of $z$ by "solving for" $c$ formally

$$c(z) = \frac{1 - \sqrt{1 - 4z}}{2z}.$$

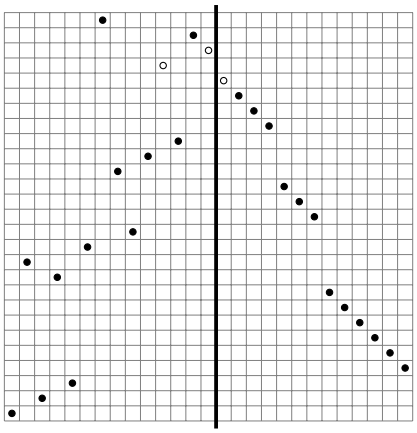
## 2.3 Enumeration

Let $P \in \mathrm{Av}(abc)$ and let $\mathcal{P}$ be the corresponding Dyck path, and let $X \in \mathrm{Av}(xy)$. Recall that when enumerating elements of $\mathrm{Av}(abc|xy)$, we chose to place the vertical gridline as far right as possible. If it was any further right, there would be a point on the RHS which would serve as $c$ in a copy of $abc$. In Figure 2.5b, for instance, there is no copy of 231 on the LHS, but if the vertical gridline was shifted one place to the right, the three unfilled points would form a copy of 231 on the LHS of the gridline. The gridline is already as far right as possible. The permutations gridded this way are exactly the $\mathrm{Av}(abc|xy)$-griddable permutations and we are free to enumerate these gridded permutations. The setup will be as in Figure 2.5. A Dyck path $\mathcal{P}$ on the left decorated by sequences of points from the right. We will always specify whether a sequence of points corresponding to a vertical step $V$ in $\mathcal{P}$ is placed immediatelly below $V$ or immediatelly above $V$. Figure 2.5a shows the "below"case. Every griddable permutation from the juxtaposition class has a unique gridding that maximizes the size of the $\mathrm{Av}(abc)$ class, i.e. pushing the gridline as far to the right as possible. We call the leftmost point on the RHS the *first* point on the RHS. We say that a point $r$ on the RHS is *enclosed* by two points $p, q$ on the LHS if $p$ is above $r$ and $r$ is above $q$ in the drawing of the juxtaposition. If $\mathcal{D}'$ is a section of a Dyck path $\mathcal{D}$ on the LHS, we say that a point $r$ on the RHS is *contained* in $\mathcal{D}'$ whenever the end points of $\mathcal{D}'$ enclose $r$. A *Catalan block* $\mathcal{D}'$ (or a *Dyck block*) is a section of a Dyck path $\mathcal{D}$ which is a Dyck path with respect to some offset of the diagonal (it begins and ends on it and stays on one side of it at all times). Notice that a Dyck block is

essentially a collection of consecutive excursions that all have the same offset from the diagonal.



(a) Sequences of points are placed at the bottom of the corresponding vertical steps (highlighted).



(b) Gridline is pushed as far right as possible: the distinguished points would form a copy of 231 on the LHS.

Figure 2.5: On the left is a decorated Dyck path of a 231-avoider, while on the right is the corresponding gridded permutation from Av(231|12). Every griddable permutation in Av(231|12) has exactly one such gridding.

#### 2.3.0.1 Class $\mathbf{A} = \mathrm{Av}(231|12)$

As we mentioned already, a symmetry of this class was enumerated by Bevan [Bev17] as a step towards the enumeration of $\mathrm{Av}(4213, 2143)$, by exploiting a tree-like structure of the permutations in $\mathrm{Av}(231)$. Here, we enumerate $\mathrm{Av}(231|12)$ by first describing through a context-free grammar.

We represent a 231-avoiding permutation $P$ by a Dyck path $\mathcal{P}$ from top-right to bottom-left, and below the diagonal. Sequences of points (each possibly empty) on the RHS are placed immediately below the corresponding down steps. Again, the gridding of $P \in \mathrm{Av}(231|12)$ is chosen to maximize the number of points on the LHS. Therefore, the leftmost point on the RHS must be below the "1" in the topmost 12 on the LHS. In a corresponding Dyck path $\mathcal{P}$, this means that we can start placing point sequences on the RHS as soon as we have encountered a down step ($\mathsf{D}$) in $\mathcal{P}$ that has a left step ($\mathsf{L}$) before it. The following letters will be needed to build $\mathcal{P}$.

$\mathsf{L}$ – left step

$\mathsf{D}$ – down step before any left steps occured

$\mathbf{D}$ – down step after left step already occured

We denote by $\mathbf{C}$ a Dyck path over letters $\mathsf{L}$ and $\mathbf{D}$, while $\mathsf{C}$ is a standard Dyck path over $\mathsf{L}$ and $\mathsf{D}$. Given these building blocks, a Dyck path representing a 231-avoider can only start with a $\mathsf{D}$, followed by a possibly empty Dyck path, and returning to the diagonal with a left-step $\mathsf{L}$. Afterwards any other Dyck path can be appended as long as it is over letters $\mathsf{L}, \mathbf{D}$. Hence, the CFG describing the elements of Av(231|12) has the following rules.

$$\mathsf{S} \to \epsilon \mid \mathsf{DSL}\mathbf{C}$$
$$\mathbf{C} \to \epsilon \mid \mathbf{D}\mathsf{C}\mathsf{L}\mathbf{C}$$

From here we can directly pass to the functional equations by letting $z$ track the down steps (whether $\mathsf{D}$ or $\mathbf{D}$) and $t$ track the sequences of points in the right compartment of the grid. In particular, $\mathbf{D}$ transcribes as $tz$. Both $s$ and $\mathbf{c}$ are functions of $t$ and $z$.

$$s = 1 + zs\mathbf{c}$$
$$\mathbf{c} = 1 + tz\mathbf{c}^2$$

Since there is no ambiguity to how we place points in the right compartment of the grid, only decreasing sequences are allowed, we let $t = 1/(1-z)$. Solving for $s$ gives the next theorem.

**Theorem 2.3.1** (Bevan [Bev17]). *The generating function for* $\mathbf{A} = \mathrm{Av}(231|12)$ *is*

$$s_{\mathbf{A}}(z) = \frac{1 + z - \sqrt{1 - 6z + 5z^2}}{2z(2-z)}$$

The generating function $s_{\mathbf{A}}(z)$ stores terms of A033321 in OEIS [Inca]. The initial twelve coefficients of $s_{\mathbf{A}}(z)$ are 1, 2, 6, 21, 79, 311, 1265, 5275, 22431,

96900, 424068, 1876143. Again, see Bevan [Bev17] for a different approach to enumerating this juxtaposition class. Also, Miner [Min16] recently enumerated Av(321|12) (via yet another approach), which is in bijection with Av(231|12) as we show in Section 2.4.

#### 2.3.0.2 Class B = Av(321|21)

We represent a 321-avoiding permutation $P$ by a Dyck path $\mathcal{P}$ from the bottom left to the top right corner of the grid, and staying above the diagonal. The sequences of points (each possibly empty) on the RHS, associated with vertical steps (up steps) on the LHS, are always placed directly below the corresponding **U** on the LHS (i.e. below the point in $P$ associated with the **U**). Additionally, we need to correct for the points on the RHS which can occur above everything on the LHS. We do this when transcribing the grammar to equations. Below is the alphabet used for our CFG.

U – up step, no sequence of points on the RHS associated with it

R – right step

**U** – up step with a possible sequence of points right below it on RHS

$\mathbf{U}_1$ – up step associated with the first point on the RHS

The CFG rules are below. Starting state is S.

$$\begin{aligned}
\mathsf{S} &\to \mathsf{C} \mid \mathsf{CM}\mathbf{C} \\
\mathsf{M} &\to \mathsf{UCM}\mathbf{C}\mathsf{R} \mid \mathbf{U}_1\mathbf{C}^{+}\mathsf{R} \mid \mathsf{B}\mathbf{E}\mathbf{U}\mathbf{C}^{+}\mathsf{R} \\
\mathsf{B} &\to \mathbf{U}_1\mathsf{R} \mid \mathsf{UCB}\mathbf{E}\mathsf{R} \\
\mathsf{C} &\to \epsilon \mid \mathsf{UCRC} \\
\mathbf{C} &\to \epsilon \mid \mathbf{UCRC} \\
\mathbf{C}^{+} &\to \mathbf{UCRC} \\
\mathbf{E} &\to \epsilon \mid \mathbf{URE}
\end{aligned}$$

We now describe the rules above in detail.

$\mathsf{S}$ – Start rule. The entire object is either a Catalan object without points on the RHS, or there is at least one point on the RHS. In the latter case, the entire word starts with a Catalan object $\mathsf{C}$, then there is the middle part $\mathsf{M}$ containing $\mathsf{U}_1$ and a 21 above this $\mathsf{U}_1$. All this is followed by a possibly empty Catalan object $\mathbf{C}$ that allows points on the RHS.

$\mathsf{M}$ – Middle part (possibly multiple excursions) with $\mathsf{U}_1$ and the first 21 above this $\mathsf{U}_1$. Part $\mathsf{M}$ either starts with a point on the RHS and then $\mathsf{U}_1\mathbf{C}^+\mathsf{R}$ guarantees a 21 in the same Catalan block that sits on the diagonal (the $\mathsf{U}_1$ is not in the corner). Or we first see the initial point on the RHS without a 21 above it and in the same Catalan block (call this block $B$) but we then see a 21 later (in this case, the $\mathsf{U}_1$ is in the corner, i.e. not in $\mathsf{U}_1\mathsf{R}$). This is the case $\mathsf{B}\mathbf{E}\mathbf{U}\mathbf{C}^+\mathsf{R}$. Or we offset the diagonal and repeat the process, i.e. we recur $\mathsf{M}$. That is, $\mathsf{UCM}\mathbf{C}\mathsf{R}$.

$\mathsf{B}$ – An excursion with the first point on the RHS and no 21 above it in the same excursion. Either we are in the base case, $\mathsf{U}_1\mathsf{R}$, or we recur the construction of $\mathsf{B}$. That is, nothing before $\mathsf{B}$ can contain a point on the RHS, and nothing after $\mathsf{B}$ can contain a 21. We get $\mathsf{UCB}\mathbf{E}\mathsf{R}$ as desired.

$\mathsf{C}$ – Catalan block without restrictions.

$\mathbf{C}$ – Catalan block with points allowed on the RHS.

$\mathbf{E}$ – Stairs with points allowed on the RHS (repeated $\mathbf{U}\mathsf{R}$).

$\mathbf{C}^+$ – Catalan block that is not empty.

We obtain the following equations from the rules above. The additional $t$ at the end of the first equation tracks the gap on the RHS above everything on the LHS. This gap could contain a non-empty increasing sequence of points.

$$\begin{aligned} s &= c + cm\mathbf{c}t \\ b &= z^2t + zcb\mathbf{e} \\ m &= zcm\mathbf{c} + b\mathbf{e}zt(\mathbf{c}-1) + z^2t(\mathbf{c}-1) \end{aligned}$$

$$c = 1 + zc^2$$
$$\mathbf{c} = 1 + zt\mathbf{c}^2$$
$$\mathbf{e} = 1 + zt\mathbf{e}$$

Given the role of $t$, to track sequences of points whose position is determined, we can again replace it with $1/(1-z)$. After the substitution and after solving the above equations, we find the following theorem.

**Theorem 2.3.2.** *The generating function for* $\mathbf{B} = \mathrm{Av}(321|21)$ *is*

$$s_{\mathbf{B}}(z) = -\frac{1 - \sqrt{1-4z} + z(-4 + \sqrt{1-4z} + \sqrt{1-5z}/\sqrt{1-z})}{2z^2}$$

The generating function $s_{\mathbf{B}}(z)$ stores the sequence A278301 of OEIS [Inca]. The initial twelve coefficients of $s_{\mathbf{B}}(z)$ are 1, 2, 6, 23, 98, 434, 1949, 8803, 39888, 181201, 825201, 3767757.

#### 2.3.0.3 Class $\mathbf{C} = \mathrm{Av}(312|21)$

We represent a 312-avoiding permutation $P$ by a Dyck path $\mathcal{P}$ starting in the bottom-left and ending in the top-right corner of the grid, and staying above the diagonal. The sequences of points (each possibly empty) on the RHS associated with $\mathsf{U}$s are placed immediately above those respective $\mathsf{U}$s. The letters in the alphabet stay the same as above when we described the class $\mathbf{B}$.

$\mathsf{U}$ – up step, no sequence of points on the RHS associated with it

$\mathsf{R}$ – right step

$\mathsf{U}$ – up step with a possible sequence of points right above it on RHS

$\mathsf{U}_1$ – first $\mathsf{U}$, marks the up step with the first point on the RHS

The following rules describe the context-free grammar enumerating Av(312|21).

$$\mathsf{S} \to \mathsf{C} \mid \mathsf{CMC}$$

$$
\begin{aligned}
\mathsf{M} &\to \mathbf{U}_1\mathbf{C}^+\mathsf{R} \mid \mathsf{UCB}\mathbf{C}^+\mathsf{R} \mid \mathsf{UCM}\mathbf{C}\mathsf{R} \\
\mathsf{B} &\to \mathbf{U}_1\mathsf{R} \mid \mathsf{UCBR} \\
\mathsf{C} &\to \epsilon \mid \mathsf{UCRC} \\
\mathbf{C} &\to \epsilon \mid \mathbf{UC}\mathsf{R}\mathbf{C} \\
\mathbf{C}^+ &\to \mathbf{UC}\mathsf{R}\mathbf{C}
\end{aligned}
$$

We now justify the above rules.

S – the entire sequence S is either a Catalan object C (no points on the RHS) or it has a distinguished middle part M which is a Catalan object sitting on the diagonal which contains the first point on the RHS. Given the way a 312-avoiding permutation sits in the Dyck path, it follows that M also contains the 21 that encloses this first point on the RHS. Such M is then followed by **C** allowing points on the RHS.

M – The middle part is either one of the two base cases, or the recursive case. In the base cases, the first point on the RHS occurs in M – either in a corner ($\mathbf{U}_1\mathsf{R}$) or not. If not in the corner, then the rule to capture this is $\mathbf{U}_1\mathbf{C}^+\mathsf{R}$ – the Catalan object must be non-empty as this part sits on the diagonal and so the point on the RHS associated with $\mathbf{U}_1$ must be enclosed by a 21 on the LHS. If the first point on the RHS occurs in the corner, then we refer to B, i.e. the rule is $\mathsf{UCB}\mathbf{C}^+\mathsf{R}$. Notice that we want the Catalan object after B to be non-empty, as B only contains R steps after a $\mathbf{U}_1$. Finally, the recursive step is $\mathsf{UCM}\mathbf{C}\mathsf{R}$ as expected.

B – this part describes the Catalan object containing $\mathbf{U}_1$ (first up-step which is immediately followed by a point on the RHS) that is in the corner – i.e. forms $\mathbf{U}_1\mathsf{R}$. Notice that B cannot sit on the diagonal, for otherwise the first point on the RHS would not be enclosed in a 21 on the LHS. So the base case for B is just a corner with a point on the RHS, and the recursive step is an object B preceded by an arbitrary Catalan object, this all offset from the start level. The way B is used inside M makes it unnecessary to have $\mathsf{UCB}\mathbf{C}\mathsf{R}$ instead of UCBR. It would lead to double-counting.

$\mathsf{C}$ – Catalan block without any restrictions.

$\mathbf{C}$ – Catalan block with arbitrary point sequences allowed on the RHS immediately after every up step on the LHS.

$\mathsf{C}^+$ – Catalan block that is not empty.

The equations that follow from the above rules are as follows (in the same order). The functions $s, m, b, \mathbf{c}, c$ take arguments $z$ and $t$. As before, set $t = 1/(1-z)$ as it represents a (possibly empty) sequence of points on the RHS.

$$\begin{aligned} s &= c + cm\mathbf{c} \\ m &= z^2t(\mathbf{c}-1) + zcb(\mathbf{c}-1) + zcm\mathbf{c} \\ b &= zcb + z^2t \\ c &= 1 + c^2z \\ \mathbf{c} &= 1 + \mathbf{c}^2zt \end{aligned}$$

**Theorem 2.3.3.** *The generating function for the class* $\mathbf{C} = \mathrm{Av}(312|21)$ *is*

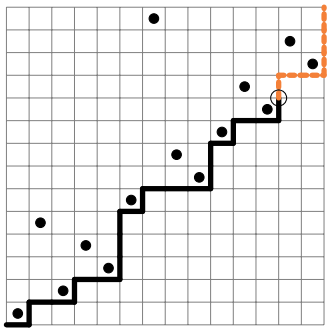
$$s_{\mathbf{C}}(z) = -\frac{(1+\sqrt{1-4z})(1-z+\sqrt{x-1}\sqrt{5x-1})}{4z}$$

The initial twelve coefficients of $s_{\mathbf{C}}(z)$ are 1, 2, 6, 22, 88, 367, 1568, 6810, 29943, 132958, 595227, 2683373. It turns out that this sequence enumerating $\mathrm{Av}(312|21)$ is the sequence A165538 in OEIS. As we mention in Section 2.2, A165538 also enumerates the class $\mathrm{Av}(4312, 3142)$, see [AAV14].

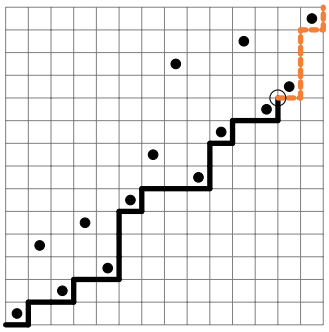
## 2.4 Bijections

Before we describe bijections between the juxtaposition classes, let us introduce the notion of an *articulation point* in 231-avoiders and 321-avoiders in order to define a canonical bijection between the two classes (in which the articulation point is fixed).

Let $P$ be a 231-avoider and $\mathcal{P}$ the corresponding Dyck path below the diagonal from the top-right corner to the bottom-left corner. The articulation point $x$ on



(a) 231-avoider.



(b) 321-avoider.

Figure 2.6: Articulation points in a 231-avoider and 321-avoider (left and right, respectively) at the position $(12, 10)$, marked by empty circles in both pictures.

$\mathcal{P}$ is any and all of the following.

- The end-point of the first down step after a left-step has occured in $\mathcal{P}$.
- The intersection of $\mathcal{P}$ and the gridline directly below the "1" in the topmost copy of 12 in $P$.

The two descriptions are readily seen equivalent from Figure 2.6a where $P$ and $\mathcal{P}$ are drawn into the same picture. It follows that an articulation point $x$ partitions $\mathcal{P}$ into $\mathcal{P}'$ (from the top to $x$) and $\mathcal{P}''$ (from $x$ to the bottom). Notice that $\mathcal{P}'$ is unique in the sense that it is the very part of a Dyck path that for which $x$ is an articulation point. In other words, there is one and only one $\mathcal{P}'$ that completes $\mathcal{P}''$ into $\mathcal{P}$. To be exact, $\mathcal{P}'$ must end with a $\mathsf{D}$ (down step) and this $\mathsf{D}$ must be the first $\mathsf{D}$ after an $\mathsf{L}$ occured. Hence, $\mathcal{P}' = \mathsf{XD}$, where $\mathsf{X}$ is a string of consecutive $\mathsf{D}$s followed by consecutive $\mathsf{L}$s.

Let $Q$ be a 321-avoider and $\mathcal{Q}$ its Dyck path below the diagonal from the top-right corner to the bottom-left corner. Then the articulation point $y$ on $\mathcal{Q}$ is defined as any/all of the following.

- It is the end-point of the first left step that does not end on the diagonal.
- It is the bottom left corner of the grid box of "1" in the topmost 21 in $Q$ which is simultaneously also a point on $\mathcal{Q}$.

Again, example is in Figure 2.6b. Therefore, $y$ partitions $\mathcal{Q}$ into $\mathcal{Q}'$ (from the top to $y$) and $\mathcal{Q}''$ (from $y$ to the bottom). Given $\mathcal{Q}''$, there is a unique $\mathcal{Q}'$ that completes it into $\mathcal{Q}$. Indeed, $\mathcal{Q}'$ must end with an $\mathsf{L}$. It must start with a sequence of $\mathsf{DL}$ pairs, followed by a sequence of $\mathsf{D}$s until the final letter $\mathsf{L}$.

By an articulation point of a permutation we mean the articulation point of the corresponding Dyck path. Next lemma tells us that we can pass between the two classes, Av(231) and Av(321), and keep the articulation point fixed.

**Lemma 2.4.1.** *There exists a bijective map* $\lambda : \mathrm{Av}(231) \to \mathrm{Av}(321)$ *that fixes articulation points.*

*Proof.* Given the position of an articulation point $x$, let $\mathcal{P}''$ and $\mathcal{Q}''$ be identical. Then there are unique completions $\mathcal{P}'$ and $\mathcal{Q}'$, of $\mathcal{P}''$ and $\mathcal{Q}''$, that make them into Dyck paths $\mathcal{P}$ and $Q$, respectively. Clearly, this is a bijection, say $\lambda'$, between Dyck paths $\mathcal{P}$ with an articulation point $x$ and Dyck paths $\mathcal{Q}$ with the same articulation point $x$. Given the bijections between 231-avoiders and Dyck paths, and 321-avoiders and Dyck paths from Section 2.2, we can define $\lambda$ as a composition of these with $\lambda'$. This makes $\lambda$ a bijection as desired.

$$\mathrm{Av}(231) \to \mathrm{Dyck}(\mathrm{Av}(231)) \xrightarrow{\lambda'} \mathrm{Dyck}(\mathrm{Av}(321)) \to \mathrm{Av}(321).$$

□

For later use, we define two more notions. Firstly, when we write $P = P_1P_2$ we mean that $P_1$ and $P_2$ to be the permutations on the LHS and the RHS of the gridding of $P$ that maximizes the size of $P_1$, i.e. if $P_1$ contained the first point of $P_2$, it would not be an *abc*-avoider. In particular, note the interplay of an articulation point and the vertical gridline. If the gridline was more to the left, then the (possibly different) articulation point would be below the first point on the RHS. Secondly, assume the down steps (vertical steps) in a Dyck path $\mathcal{P}$ corresponding to a permutation $P_1$ are labelled 1 through $n$ in order of their appearance in $\mathcal{P}$. Then we can describe the permutation $P_2$ on the RHS of the gridding by a sequence $h$ of length $n$. Given $h = (h_1, \ldots, h_n)$, $h_i$ is the length of the sequence of points associated with the $i$-th down step $D_i$. Depending on our choice (see Section 2.3), these sequences of $h_i$ points are placed either immediately

above or immediately below the point of the permutation corresponding to $D_i$. In any case, such a sequence $h$ together with a Dyck path $\mathcal{P}$ uniquely describe a permutation from one of the juxtaposition classes $\mathrm{Av}(abc|xy)$. Therefore, we can refer to a $P = P_1P_2 \in \mathrm{Av}(231|21)$ as a pair $(P, h_P)$ (or alternatively $(P_1, h_P)$) for the appropriate sequence $h_P$.

Theorem 2.4.1 and 2.4.2 both make use of the bijection $\lambda$ from Lemma 2.4.1.

**Theorem 2.4.1.** *There exists a bijection* $\theta : \mathrm{Av}(231|12) \to \mathrm{Av}(321|12)$.

*Proof.* For visual aid, see Figure 2.6. Let $(P, h_P)$ be a permutation from $\mathrm{Av}(231|12)$. We define $\theta$ to be $\lambda$ when restricted to $P_1$ and to be an identity map on $h_P$. Provided the gridding is unique for every $P \in \mathrm{Av}(231)$ and the resulting object is a permutation from $\mathrm{Av}(321|12)$, $\theta$ is a bijection. Every $\mathrm{Av}(231|12)$ griddable permutation has exactly one gridding of our choice – pushing the gridline as right as possible. Given that $\lambda$ fixes the articulation point of $P$ and $h_P$ is unchanged under $\theta$, the gridline in the image of $\theta$ is as far right as possible – making the gridding unique. Hence, $\theta : (P, h_P) \mapsto (\lambda(P), h_P)$ is a bijection from $\mathrm{Av}(231|12)$ to $\mathrm{Av}(321|12)$. □

The next bijection acts in an analogous way to $\theta$, except this time the sequence on the RHS is increasing. Therefore, we additionally need to show that there always is a point on the RHS below an articulation point.

**Theorem 2.4.2.** *There exists a bijection* $\psi : \mathrm{Av}(231|21) \to \mathrm{Av}(321|21)$.

*Proof.* Since the LHS classes in these juxtapositions are the same as those in Theorem 2.4.1, the above proof carries over to this case as it is. One only needs to notice, additionally, that $\theta$ (and hence $\psi$) fixes the articulation point and acts as identity on the RHS sequence of points. This preserves the relative vertical position of the articulation point and any point on the RHS. Therefore, if there is a point on the RHS below the articulation point in a $P \in \mathrm{Av}(231|21)$, then there is a point on the RHS and below the articulation point of $\psi(P) \in \mathrm{Av}(321|21)$. □

Unlike the bijections $\theta$ and $\psi$, the last bijection $\phi$ does not make use of Lemma 2.4.1. Instead, we reshuffle excursions in the Dyck paths and reverse the order of up steps in the middle excursion $\mathsf{M}$.

**Theorem 2.4.3.** *There exists a bijection* $\phi : \mathrm{Av}(312|21) \to \mathrm{Av}(312|12)$.

*Proof.* The idea of this bijection is to "reverse" the Dyck path on the LHS of the juxtaposition in some sensible way — this also leads to the sequence $h_P$ being reordered. Provided this is done in a reversible way, we are done. Let $P \in \mathrm{Av}(312|12)$ be $P = P_1P_2$. Let $P_1$ be represented by a Dyck path $\mathcal{P}$ and let $\mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_k$, $k \geq 1$, be the *decomposition of* $\mathcal{P}$ *into excursions*. Recall from Section 2.3.0.3 the way $\mathrm{Av}(312|21)$ was enumerated. One of the excursions in $\mathcal{P}$, say $\mathcal{P}_i$, was described by part $\mathsf{M}$ in the CFG. Construct a new Dyck path $\mathcal{Q}'$ from $\mathcal{P}$ by setting $\mathcal{Q}' = \mathcal{P}_{i+1} \oplus \cdots \oplus \mathcal{P}_n \oplus \mathcal{P}_i \oplus \mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_{i-1}$. Recall that the part $\mathcal{P}_i$ is of the form $\mathsf{U} \ldots \mathsf{U}_1 \ldots \mathsf{R}$. We can, without interfering with the structure of the excursion $\mathcal{P}_i$ write the names of the up steps in the opposite direction to obtain $\mathcal{Q}_i$: a letter from $\{\mathsf{U}, \mathsf{U}_1, \mathsf{U}\}$ at position $\ell$ in $\mathcal{P}_i$ will assume position $|\mathcal{P}_i| - \ell$ in the new $\mathcal{Q}_i$. This way, up steps and down staps remain preserved (the shapes of $\mathcal{P}_i$ and $\mathcal{Q}_i$ are the same). Given that $\mathcal{P}_i$ (hence also $\mathcal{Q}_i$) is an excursion, there is a 21 in $P_1$ to enclose the lowest (in the case of $\mathcal{Q}_i$ the highest) point on the RHS. Given the transformation from $\mathcal{P}$ to $\mathcal{Q} = \mathcal{P}_{i+1} \oplus \cdots \oplus \mathcal{P}_n \oplus \mathcal{Q}_i \oplus \mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_{i-1}$, vertical steps in $\mathcal{Q}$ are in a different order than they originally were in $\mathcal{P}$. This results in the corresponding sequence $h_Q$ being a respective reshuffle of $h_P$. It is easy to check that the permutation $(Q, h_Q)$ is in the juxtaposition class $\mathrm{Av}(312|12)$. Given that the above transformation from $\mathrm{Av}(312|21)$ to $\mathrm{Av}(312|12)$ is unambiguous (i.e. reversible), we conclude that

$$\mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_k \mapsto \mathcal{P}_{i+1} \oplus \cdots \oplus \mathcal{P}_n \oplus \mathcal{Q}_i \oplus \mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_{i-1}$$

gives rise to the bijection $\phi : \mathrm{Av}(312|21) \to \mathrm{Av}(312|12)$ such that $\phi : (P, h_P) \mapsto (Q, h_Q)$. □

Theorems 2.4.1, 2.4.2 and 2.4.3 complete the information in Table 2.1.

## 2.5 Conclusion

Table 2.1 enumerates the "simplest" grid classes that are not monotone. The next step could see Catalan class replaced by a more complicated one, or increased

number of cells. If exact enumeration is beyond reach, it would be interesting to obtain information about the generating functions of such grid classes. For instance, consider a grid class formed by taking a monotone geometric grid class and replacing one cell by a Catalan class. Does every such grid class have an algebraic generating function? Can we say anything about generating functions of similar grid classes with a non-Catalan cell?

A problem not closely related to the exploration of grid classes, but interesting nevertheless, is finding a bijection between the juxtaposition class $\mathbf{C} = \mathrm{Av}(312|21)$ and the class $\mathrm{Av}(4312, 3142)$. They happen to be enumerated by the same sequence A165538 on OEIS as we repeatedly mentioned before. See Albert, Atkinson and Vatter [AAV14] for their enumeration of $\mathrm{Av}(4312, 3142)$.

# Chapter 3

# Iterated juxtapositions

The goal of this chapter is to make further progress on describing permutation grid classes. In Chapter 2, we focus on exact enumeration of a specific set of permutation grid classes of the form $\mathcal{C}|\mathcal{M}$, where $\mathcal{C}$ is a Catalan class and for the moment $\mathcal{M}$ is a monotone class. In this chapter, we trade some amount of "exactness" for more generality. In particular, we address permutation grid classes of the form $\mathcal{M}_1|\dots|\mathcal{M}_k|\mathcal{C}|\mathcal{M}_{k+1}|,\dots|\mathcal{M}_{k+\ell}$, for some $k,\ell \geq 0$, where $\mathcal{C}$ is an arbitrary *context-free* permutation class. We define what context-free means in Section 3.1. For now, let us note that "Context-free" is significantly more general than "Catalan", yet well-behaved enough to deal with. As opposed to the Catalan classes, we do not enumerate all these context-free juxtapositions exactly as there are infinitely many of them (compare that to the six Catalan juxtapositions) and they are very disparate. In principle though, our method allows us to enumerate each one of such classes individually. However, exact enumeration is a mere side-product. Instead, we focus on proving that the permutation classes of the form $\mathcal{M}_1|\dots|\mathcal{M}_k|\mathcal{C}|\mathcal{M}_{k+1}|\dots|\mathcal{M}_{k+\ell}$ admit algebraic generating functions.

Algebraicity follows from the context-free property of the class itself. In fact, *context-freeness* is what we work with and exploit. Algebraicity of the generating function is a consequence of it. Still, notice that algebraic generating function and as "nice" as one can hope for given that many generating functions enumerating context-free classes are already algebraic and non-rational. Below is a hierarchy of families of generating functions showing that the family of algebraic power series

is one of the more special ones.

$$\text{rational} \subset \text{algebraic} \subset D\text{-finite} \subset D\text{-algebraic} \subset \text{power series}$$

From this viewpoint, our result states that by appending arbitrary but finite number of monotone classes on either side of a context-free class $\mathcal{C}$, the resulting class does not lose context-freeness and its generating function does not get qualitatively worse compared to the one enumerating $\mathcal{C}$. A notable corollary of this result is, for instance, that juxtaposition of monotone classes on either side of a context-free permutation class $\mathcal{C}$ with finitely many simple permutations admits an algebraic generating function. This is a significant subcase and context-free classes with finitely many simples were studied in e.g. []. Moreover, we work out several examples explicitly to obtain exact generating functions. These are $\mathrm{Av}(321|21)$, $\mathcal{M}|\mathcal{M}|\mathcal{M}$ from [Bev15b], and $\mathcal{S}|\mathcal{M}$ (separable next to monotone).

## 3.1 Introduction, definitions, prerequisites

To begin, we juxtapose a context-free permutation class $\mathcal{C}$ with a finite row of monotone classes $\mathcal{M}_1|\ldots|\mathcal{M}_k$ on the right. Additionally, we assume for the moment that each $\mathcal{M}_i$ is monotone and increasing. We later argue that for decreasing classes, a symmetry of our argument applies and renders the case essentially identical to increasing $\mathcal{M}$. We continue by proving that appending $\mathcal{M}$ on the left-hand side or the right-hand side does not have any bearing on our arguments. So eventually without loss of generality, we append only increasing $\mathcal{M}$ and only from the right — at least for now. As mentioned above, the work in this chapter extends the work in Chapter 2 in two directions. One, the condition that $\mathcal{C}$ is a Catalan permutation class is replaced by requiring $\mathcal{C}$ to only be context-free. Two, juxtaposition from the right is iterated a finite number of times instead of just once. Before we proceed with the statement of the result, let us set the scene. The following definition is taken from Flajolet and Sedgewick [FS09], Section I.5.4, Definition I.13.

**Definition 3.1.1** (Context-free specification)**.** A class $\mathcal{C}$ is said to be *context-free*

if it coincides with the first component $\mathcal{S}_1$ of a system of equations

$$\begin{cases} \mathcal{S}_1 &= f_1(\mathcal{Z}, \mathcal{S}_1, \ldots, \mathcal{S}_r) \\ &\vdots \\ \mathcal{S}_r &= f_r(\mathcal{Z}, \mathcal{S}_1, \ldots, \mathcal{S}_r) \end{cases} \tag{3.1}$$

where each $f_i$ is a constructor that only involves operations of combinatorial sum $(+)$ and cartesian product $(\times)$, as well as the neutral/empty class $\mathcal{E} = \{\emptyset\}$.

For our purposes, two classes are the same if there exists a bijection between them. The following definition implies this.

**Definition 3.1.2.** Two combinatorial classes are *combinatorially isomorphic* if and only if their counting sequences are identical. This is equivalent to existence of a size-preserving bijection between the two classes.

Before we proceed to modify the definition of a context-free class, we state the key result of Chomsky and Schutzenberger as Theorem 3.1.1. It relates the character of the class' context-free combinatorial specification to the character of the class' generating function. If the former is "nice", then so is the latter. Our work then consists of proving that the juxtaposition classes in question have "nice" combinatorial specifications. Once that is known, Theorem 3.1.1 translates the result into the language of generating functions as we state it later.

**Theorem 3.1.1** (Chomsky-Schutzenberger, Proposition I.7. in [FS09])**.** *A combinatorial class $\mathcal{C}$ that is context-free admits an ordinary generating function that is an algebraic function. In other words, there exists a (non-null) bivariate polynomial $P(z, y) \in \mathbb{C}[z, y]$ such that*

$$P(z, C(z)) = 0.$$

As mentioned, we are going to need a modification of the definition of a context-free class that will allow us to decorate context-free permutation classes.

**Definition 3.1.3** (Tracking the left-most and the right-most points)**.** We say that a context-free class specification of $\mathcal{C}$ *tracks the right-most and the left-most point*

*by vertical position* if it is combinatorialy isomorphic to the context-free class with the specification $\mathscr{S}$ below and all Cartesian products in $\mathscr{S}$ are recorded left-to-right as they occur bottom-to-top in $\mathcal{C}$. The asterisk ($^*$) [circle ($^\circ$)] in $\mathscr{S}$ mark the right-most [left-most] point, or the block which contains the right-most [left-most] point inside $\mathcal{C}_i^*$ [$\mathcal{C}_i^\circ$]. Notice that a class can contain both the left-most and the right-most point, e.g. $\mathcal{C}^{\circ *}$. Analogously, we use four different atoms to describe permutation points: the usual point $\mathcal{Z}$, the rightmost point $\mathcal{Z}^*$, the leftmost point $\mathcal{Z}^\circ$, and the rightmost point that is also leftmost, $\mathcal{Z}^{\circ *}$.

$$\mathscr{S} = \left\{ \begin{array}{ll} \mathcal{C}_0^* & = f_0^*(\mathcal{Z}, \mathcal{Z}^*, \mathcal{C}_i, \mathcal{C}_i^*) \\ & \vdots \\ \mathcal{C}_r^* & = f_r^*(\mathcal{Z}, \mathcal{Z}^*, \mathcal{C}_i, \mathcal{C}_i^*) \\ \mathcal{C}_0^\circ & = f_0^\circ(\mathcal{Z}, \mathcal{Z}^\circ, \mathcal{C}_i, \mathcal{C}_i^\circ) \\ & \vdots \\ \mathcal{C}_r^\circ & = f_r^\circ(\mathcal{Z}, \mathcal{Z}^\circ, \mathcal{C}_i, \mathcal{C}_i^\circ) \\ \mathcal{C}_0^{\circ *} & = f_0^{\circ *}(\mathcal{Z}, \mathcal{Z}^*, \mathcal{Z}^\circ, \mathcal{C}_i, \mathcal{C}_i^*, \mathcal{C}_i^\circ, \mathcal{C}_i^{\circ *}) \\ & \vdots \\ \mathcal{C}_r^{\circ *} & = f_r^{\circ *}(\mathcal{Z}, \mathcal{Z}^*, \mathcal{Z}^\circ, \mathcal{C}_i, \mathcal{C}_i^*, \mathcal{C}_i^\circ, \mathcal{C}_i^{\circ *}) \\ \mathcal{C}_0 & = f_0(\mathcal{Z}, \mathcal{C}_0, \ldots, \mathcal{C}_r) \\ & \vdots \\ \mathcal{C}_r & = f_r(\mathcal{Z}, \mathcal{C}_0, \ldots, \mathcal{C}_r) \end{array} \right\} \tag{3.2}$$

where $0 \leq i \leq r$.

If a class $\mathcal{C}^*$ tracks the right-most point as outlined above, we refer to $\mathcal{C}^*$ as a *starred class*. Similarly, $\mathcal{Z}^*$ is a *starred point*, or simply the right-most point. Analogously, we have a *circled class* $\mathcal{C}^\circ$ and a *circled point* $\mathcal{Z}^\circ$. On the other hand, we will also use the terms *starless class* or *starless point* to refer to a class without a star/asterisk. For example, $\mathcal{C}, \mathcal{C}^\circ$ and $\mathcal{Z}, \mathcal{Z}^\circ$. Given the above definition, we assume from now on that every permutation in class $\mathcal{C}$ is context-free and admits a specification that tracks the right-most point. When we need to track the left-

most point we declare it explicitly.

The Cartesian product is naturally non-commutative which conveniently helps us to keep track of vertical positions. Therefore, requiring that the combinatorial specification keeps track of the rightmost point by its value amounts to merely picking a specific combinatorial specification according to how it happens to encode the right-most point in $\mathcal{C}$.

Let us consider an example of how vertical order translates into left-to-right order in the Cartesian product. For instance, $\mathcal{ZCCD}$ refers to a term which has a single point at the bottom, then somewhere above it (and to the left or to the right of it) there is an element from $\mathcal{C}$, then another element of $\mathcal{C}$ is further above the previous one, and above all of this there is an element from $\mathcal{D}$. Schematically, it could look something like the class in Figure 3.1.

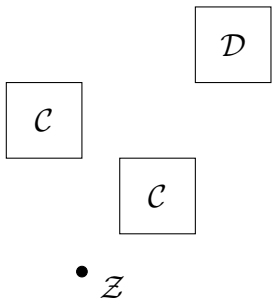



Figure 3.1: An example of a class which would correspond to a term $\mathcal{ZCCD}$ in a combinatorial specification that preserves the vertical (bottom-to-top) order of elements.

**Example 3.1.1.** The following is a context-free specification for the class of separable permutations (non-empty):

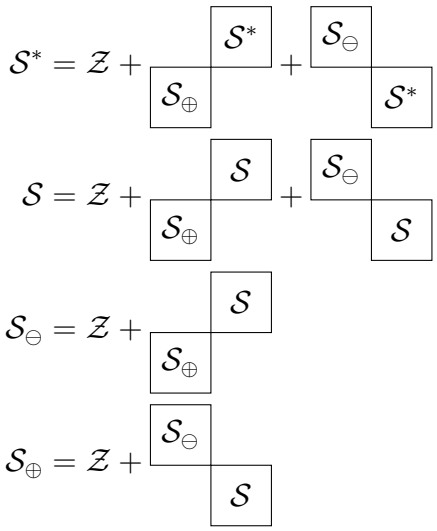
$$\begin{aligned}\mathcal{S} &= \mathcal{Z} + \mathcal{S}_{\oplus}\mathcal{S} + \mathcal{S}_{\ominus}\mathcal{S}\\ \mathcal{S}_{\ominus} &= \mathcal{Z} + \mathcal{S}_{\oplus}\mathcal{S}\\ \mathcal{S}_{\oplus} &= \mathcal{Z} + \mathcal{S}_{\ominus}\mathcal{S},\end{aligned}$$

where we use $\mathcal{S}_{\ominus}$ and $\mathcal{S}_{\oplus}$ to denote the skew-indecomposable and sum-indecomposable permutation classes. Now, among all context-free specifications of $\mathcal{S}$, we insist on

picking the following one to track the rightmost point:

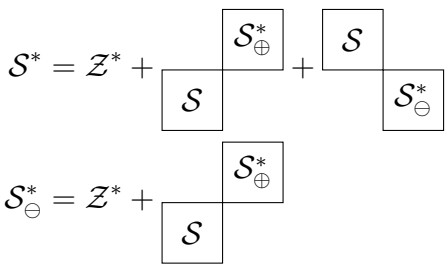
$$\begin{aligned}
\mathcal{S}^* &= \mathcal{Z}^* + \mathcal{S}_\oplus \mathcal{S}^* + \mathcal{S}^* \mathcal{S}_\ominus \\
\mathcal{S} &= \mathcal{Z} + \mathcal{S}_\oplus \mathcal{S} + \mathcal{S}\mathcal{S}_\ominus \\
\mathcal{S}_\ominus &= \mathcal{Z} + \mathcal{S}_\oplus \mathcal{S} \\
\mathcal{S}_\oplus &= \mathcal{Z} + \mathcal{S}\mathcal{S}_\ominus.
\end{aligned} \tag{3.3}$$

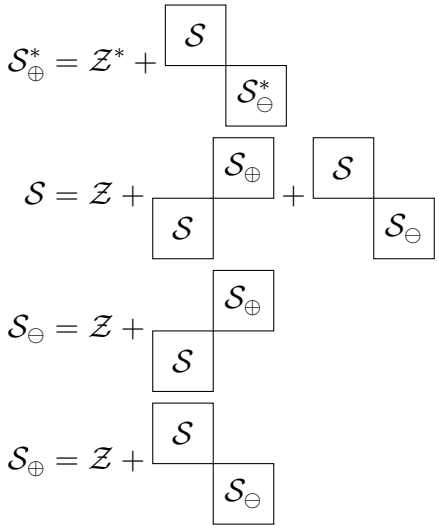
In (3.3), the class of separable permutations is represented according to the decomposition in (3.4), encoding vertical order as left-to-right. Notice, in particular, that there are multiple combinatorial specifications of $\mathcal{S}$ that track the right-most points. The one that we use is below.




(3.4)

Another combinatorial specification of $\mathcal{S}$ that tracks the rightmost point is the following one.






You will have noticed that we needed to define $\mathcal{S}^*_\ominus$ and $\mathcal{S}^*_\oplus$ in order for the specification to be closed. That is why, in Section 3.3.3, we prefer to work with (3.4) instead.

Moving from Example 3.1.1 back to the general setting, let

$$\mathcal{V} = \{\mathcal{Z}, \mathcal{Z}^*, \mathcal{C}_0, \ldots, \mathcal{C}_r, \mathcal{C}_0^*, \ldots, \mathcal{C}_r^*, \mathcal{C}_0^\circ, \ldots, \mathcal{C}_r^\circ, \mathcal{C}_0^{\circ *}, \ldots, \mathcal{C}_r^{\circ *}\}$$

be the collection of all variables occurring in the polynomials $f_i, f_i^*, f^\circ$, and $f^{\circ *}$. When we do not care to distinguish between $f_i$ and $f_j$ or $f_i, f_i^*, f^\circ$, and $f^{\circ *}$, we simply write $f$ for a polynomial in variables from $\mathcal{V}$. Similarly, when we do not distinguish between two variables in $\mathcal{V}$, we simply write $X \in \mathcal{V}$. As we just mentioned, each $f$ is a polynomial. We refer to its terms by $\mathcal{T}_h$ as in $f = \sum_{\ell=0}^{N} \mathcal{T}_h$. Each term $\mathcal{T}_h$ is a product of the variables in $\mathcal{V}$ and for a function $f^*$, each $\mathcal{T}_h$ contains exactly one starred variable (there is exactly one rightmost point in each term). The same holds for $f^\circ$.

One of the problems of enumerating permutation grid classes is that there are often multiple legal ways to grid a permutation in a given grid class. Therefore, one of the key issues we need to address is to choose a particular gridding. We represent every griddable permutation from a juxtaposition by a unique gridded version of it. We pick the gridded version that maximises the element on the right-hand side (RHS) of the juxtaposition. The following convention makes this

concept explicit. See also Figure 3.2 for an illustration.

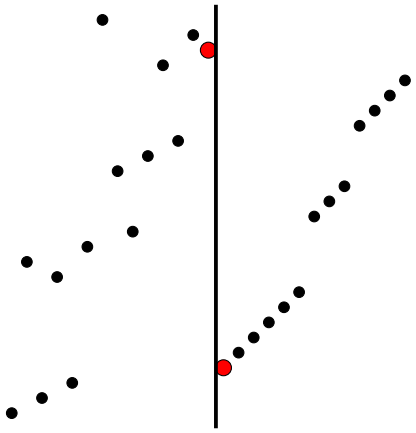


Figure 3.2: On the LHS is a permutation from $\mathcal{C}$ while on the RHS is a monotone increasing permutation. The gridline is as far left as possible: if it were shifted further to the left, the red points would form a copy of 21 on the RHS.

**Convention:** *Let $P$ be a permutation from $\mathcal{C}_1|\mathcal{C}_2$. The gridline in $P$ is chosen to be the left-most possible. I.e. if it was any further left, the sub-permutation to the right of it would not belong to the designated class $\mathcal{C}_2$.*

The Convention above is a direct consequence of the leftmost gridding. Indeed, let $\mathcal{C}|\mathcal{M}$ be a juxtaposition of any permutation class $\mathcal{C}$ with a monotone increasing permutation class $\mathcal{M}$. Given that the gridline is pushed as far left as possible, the rightmost point in the left cell must be above the leftmost point in the right cell. Otherwise, the gridline could be shifted one more point further left. Figure 3.2 shows this well. The left red point must be above the right left point, otherwise the gridline would not have been between these two red points.

Although this will be stated later when needed for Proposition 3.2.3, dealing with juxtapositions from both sides is quite natural. Given a permutation $P$ from $\mathcal{M}_1|\dots|\mathcal{M}_k|\mathcal{C}|\mathcal{M}_{k+1}|\dots|\mathcal{M}_{k+\ell}$, we first grid $\ell$ monotone classes from the right according to the Convention. Then flip the picture around a vertical axis and grid the $k$ monotone classes from the right (they used to be on the left), again according to the Convention. Of course, after the flip, we need to treat increasing $\mathcal{M}_i$ as a decreasing class, and vice versa. At the end, the leftover middle part belongs to $\mathcal{C}$.

We make some further remarks about the way we represent permutations in a juxtaposition. Let $x, y$ be two vertically consecutive points on the left-hand side of the juxtaposition $\mathcal{C}_1|\mathcal{C}_2$. An object on the RHS (e.g. a sequence of points if $\mathcal{C}_2 = \mathcal{M}$) is said to be *associated with* $x$ if it is in the horizontal section of the RHS that falls below $x$ and above $y$. If $x$ is the bottom most point on the LHS, then everything below it on the RHS is associated with $x$. See Figure 3.3 for an example.

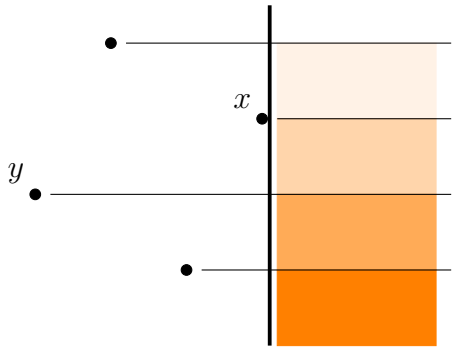



Figure 3.3: The shaded regions on the RHS each correspond to a gap between two vertically consecutive points on the LHS. The part of the right-hand side associated with $x$ is the second least opaque region.

Juxtapositions will be expressed via application of operators to context-free classes. We are going to need the following operators: $\Omega_0, \Omega_1, \Omega_\infty, \Omega_{10}, \Omega_{01}, \Omega_{11}$. They represent various forms/stages of juxtaposing a monotone class next to $\mathcal{C}$, in other words, $\Omega_i : \mathcal{C} \mapsto \mathcal{C}'$, where both $\mathcal{C}$ and $\mathcal{C}'$ are context-free permutation classes and $\mathcal{C}'$ is $\mathcal{C}$ with some kind of monotone class next to it on the right. It is important that the property of being context-free is invariant under these operators. Before we present the properties of $\Omega$ operators, we give a short conceptual description of each of them. All of them juxtapose some form of a monotone increasing class to the right of an object (whether a class or a point).

$\Omega_0$ : juxtaposes an empty permutation to the right of a class or a point

$\Omega_1$ : juxtaposes a single point to the right of a class or a point

$\Omega_\infty$ : juxtaposes an increasing sequence (possibly empty) to the right of a class or a point

$\Omega_{10}$ : juxtaposes an increasing sequence with at least the bottom point, to the right of a class or a point

$\Omega_{01}$ : juxtaposes an increasing sequence with at least the top point, to the right of a class or a point

$\Omega_{11}$ : juxtaposes an increasing sequence with distinct top and bottom points, to the right of a class or a point

Furthermore, it is desirable that all $\Omega$ operators respect the rightmost point of the class $\mathcal{C}$ they are acting on according to the following rules.

1. If the operator's code begins with 1, namely $\Omega_1, \Omega_{10}$, and $\Omega_{11}$, then the operator can only be applied to a starred class, or alternatively, to a starless class occuring before (in left-to-right order) the starred class in the Cartesian product.

2. If the operator's code ends with 0 or $\infty$, namely $\Omega_0, \Omega_{10}, \Omega_\infty$, then its output is starless. If the operator's code ends with 1, namely $\Omega_1, \Omega_{01}, \Omega_{11}$, then every term of the output contains exactly one rightmost point (a starred class or point).

Rules 1 and 2 capture the observations that:

1. If we juxtapose a monotone increasing class next to any class $\mathcal{C}$ to obtain $\mathcal{C}|\mathcal{M}$, then the leftmost/lowest point on the RHS must be below the rightmost point on the LHS. This follows from the Convention as already discussed and even pictured in Figure 3.2.

2. Juxtaposing a class on the right sometimes takes over the rightmost point from the class on the left. See figures associated with definitions of $\Omega$-operators for examples of when that happens.

Recall that $X_i$ are variables from $\mathcal{V}$. We view every $f$ as a finite sum of terms $\mathcal{T}_h$, each of which is a product of $X_i$s, i.e. $\mathcal{T}_h = X_1 X_2 \cdots X_m$ for some $m = m(h)$, with all $X_i \in \mathcal{V}$. Without loss of generality, we let $k \in [m]$ be the index of a starred class, i.e. $X_k^*$. In the forthcoming definitions of $\Omega$ operators, let $\mathcal{C}$ be a

permutation class that admits a combinatorial specification (combinatorially) isomorphic to $\mathscr{S}$ in (3.2).

**Operator $\Omega_0$**

This operator juxtaposes a class (starred or not) with an empty sequence on the right. Notice, in particular, that $\Omega_0$ distributes over both $+$ and $\times$, and that it erases $^*$.

$$\begin{aligned} \Omega_0(\mathcal{Z}) &= \mathcal{Z} \\ \Omega_0(\mathcal{Z}^*) &= \mathcal{Z} \\ \Omega_0(\mathcal{Z}^\circ) &= \mathcal{Z}^\circ \\ \Omega_0(\mathcal{Z}^{\circ *}) &= \mathcal{Z}^\circ \\ \Omega_0(\mathcal{T}_h) = \Omega_0(X_1 \cdots X_m) &= \Omega_0(X_1)\Omega_0(X_2 \cdots X_m) \end{aligned} \tag{3.5}$$

**Operator $\Omega_\infty$**

This operator juxtaposes a class (starred or not) with a monotone increasing class – possibly empty. Again, $\Omega_\infty$ is distributive over operations $+$ and $\times$. As $\Omega_0$, it also erases the $^*$. We get the following definition of $\Omega_\infty$. Consult Figure 3.4 with this definition.

$$\begin{aligned} \mathcal{M} &= \mathcal{Z} + \mathcal{M}\mathcal{Z} \\ \Omega_\infty(\mathcal{Z}) &= \mathcal{M} \\ \Omega_\infty(\mathcal{Z}^*) &= \mathcal{M} \\ \Omega_\infty(\mathcal{Z}^\circ) &= (\mathcal{M} + \mathcal{E})\mathcal{Z}^\circ \\ \Omega_\infty(\mathcal{Z}^{\circ *}) &= (\mathcal{M} + \mathcal{E})\mathcal{Z}^\circ \\ \Omega_\infty(\mathcal{T}_h) &= \Omega_\infty(X_1)\Omega_\infty(X_2 \cdots X_m) \end{aligned} \tag{3.6}$$

We only included $\mathcal{M}$ to keep the definition self-contained.

**Operator $\Omega_1$**

This operator juxtaposes a class (starred or not) with a single point. It turns out that $\Omega_1$ is linear (over $+$), but does not distribute over $\times$. It either introduces or

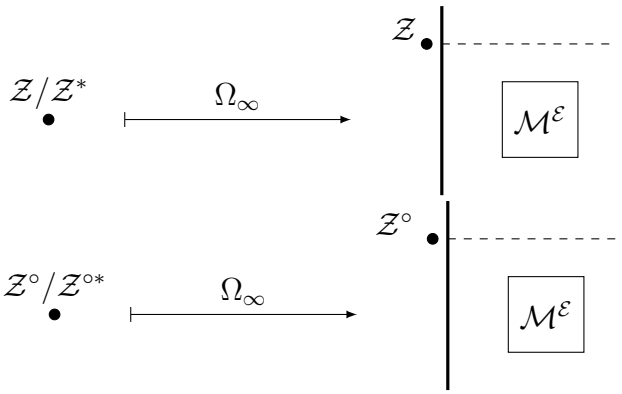



Figure 3.4: $\Omega_\infty$: Since $\Omega_\infty$ erases stars, $\mathcal{Z}$ and $\mathcal{Z}^*$ are mapped to the same object. However, $\mathcal{Z}^\circ$ is preserved under all $\Omega$ operators. We denote $\mathcal{M}+\mathcal{E}$ by $\mathcal{M}^{\mathcal{E}}$.

relocates the right-most point $^*$.

$$\begin{aligned}
\Omega_1(\mathcal{Z}) &= \mathcal{Z}^*\mathcal{Z} \\
\Omega_1(\mathcal{Z}^*) &= \mathcal{Z}^*\mathcal{Z} \\
\Omega_1(\mathcal{Z}^\circ) &= \mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_1(\mathcal{Z}^{\circ *}) &= \mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_1(\mathcal{T}_h) &= \begin{cases} \Omega_1(X_1^*)\Omega_0(X_2\cdots X_m) & \text{if } k=1 \\ \Omega_1(X_1)\Omega_0(X_2\cdots X_m)+\Omega_0(X_1)\Omega_1(X_2\cdots X_m), & \text{if } k>1. \end{cases}
\end{aligned} \tag{3.7}$$

The base cases ($\mathcal{Z}, \mathcal{Z}^*$ and $\mathcal{Z}^\circ$) are drawn in Figure 3.5. The recursive step $\mathcal{T}_h$ consists of two cases. Either the bottom-most class/point $X_1$ in $\mathcal{T}_h$ is starred (i.e. $X_1 = X_1^*$ or $X_1 = X_1^{\circ *}$) or not. If $X_1^*/X_1^{\circ *}$, then $\Omega_1$ must be applied to it (as it must be applied to a starred class/point or a class/point below it) and $\Omega_0$ is applied to the rest of the classes, $\Omega_0(X_2\cdots X_m)$. If the first class (or point) $X_1$ is not starred, then there are two options. Either apply $\Omega_1$ to $X_1$ and $\Omega_0$ to $X_2\cdots X_m$, or apply $\Omega_0$ to $X_1$ and recursively apply $\Omega_1$ to $X_2\cdots X_m$. Defining operators recursively will be useful when we apply them to permutation classes iteratively.

**Operator $\Omega_{01}$**

This operator juxtaposes a class $\mathcal{C}$ with a monotone sequence that has a distinguished right-most (top-most) point but not the left-most (bottom-most) point. Here, distinguished simply means that we can isolate it from the rest of the points

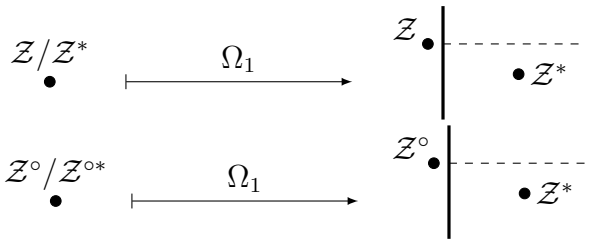



Figure 3.5: $\Omega_1$: Juxtaposing a single point to the right of any point means that the original point is not the right-most anymore and the new one becomes right-most. The left-most point is unafected.

and treat it separately. Recall that this was not the case for $\Omega_\infty$. As usually, $\Omega_{01}$ is linear. It does not distribute over the Cartesian product. Also, $\Omega_{01}$ introduces the right-most point if there was none in $\mathcal{C}/\mathcal{C}^\circ$. Like the rest of the operators, $\Omega_{01}$ does not interact with the left-most point in $\mathcal{C}^\circ/\mathcal{C}^{\circ *}$. See Figure 3.6.

$$\begin{aligned}
\Omega_{01}(\mathcal{Z}) &= (\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} \\
\Omega_{01}(\mathcal{Z}^*) &= (\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} \\
\Omega_{01}(\mathcal{Z}^\circ) &= (\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_{01}(\mathcal{Z}^{\circ *}) &= (\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_{01}(\mathcal{T}_h) &= \Omega_{01}(X_1)\Omega_0(X_2\cdots X_m) + \Omega_\infty(X_1)\Omega_{01}(X_2\cdots X_m)
\end{aligned} \tag{3.8}$$

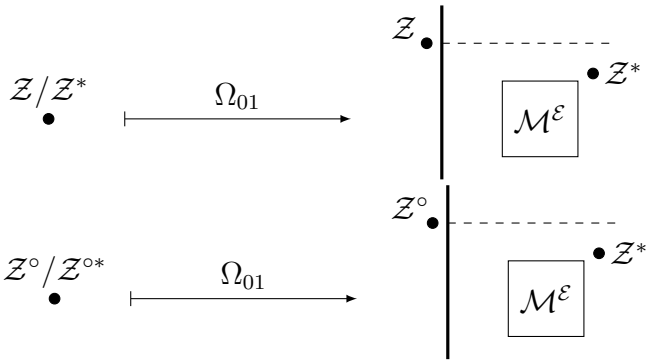



Figure 3.6: $\Omega_{01}$: Juxtaposing a monotone sequence with tracked right-most point to the right of a point renders that point not right-most (regardless of whether it was or was not before). The right-most point is now the right-most point on the RHS. Also, $\Omega_{01}$ does not affect whether the original point is left-most. We denote $\mathcal{M}+\mathcal{E}$ by $\mathcal{M}^\mathcal{E}$.

**Operator $\Omega_{10}$:**

This operator juxtaposes a class $\mathcal{C}$, which might or might not track the left-most and right-most points, with a monotone sequence on the right which has a distinguished left-most (bottom-most) point. Again, $\Omega_{10}$ is linear over $+$. It erases the right-most point from $\mathcal{C}^*/\mathcal{C}^{\circ *}$ and does not affect the left-most point in $\mathcal{C}^\circ/\mathcal{C}^{\circ *}$. The base cases of the definition below are described in Figures 3.7.

$$\begin{aligned}
\Omega_{10}(\mathcal{Z}) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z} \\
\Omega_{10}(\mathcal{Z}^*) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z} \\
\Omega_{10}(\mathcal{Z}^\circ) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^\circ \\
\Omega_{10}(\mathcal{Z}^{\circ *}) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^\circ \\
\Omega_{10}(\mathcal{T}_h) &= \begin{cases} \Omega_{10}(X_1^*)\Omega_\infty(X_2\cdots X_m) & \text{if } k=1 \\ \left.\begin{array}{l}\Omega_{10}(X_1)\Omega_\infty(X_2\cdots X_m)+ \\ +\Omega_0(X_1)\Omega_{10}(X_2\cdots X_m)\end{array}\right\} & \text{if } k>1. \end{cases}
\end{aligned} \tag{3.9}$$

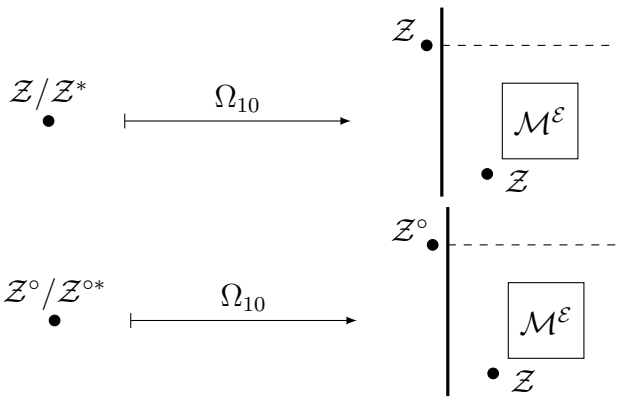



Figure 3.7: $\Omega_{10}$: Juxtaposing a monotone sequence with tracked left-most point to the right of a point renders that point not right-most (regardless of whether it was or was not before) and does not affect whether that point is left-most. We denote $\mathcal{M}+\mathcal{E}$ by $\mathcal{M}^{\mathcal{E}}$.

**Operator $\Omega_{11}$:** This operator juxtaposes a class $\mathcal{C}$, which might and might not track the right-most and left-most points, on the right with a monotone increasing sequence that has both its extremal points distinguished: the left-most point and the rightmost point. As usual, $\Omega_{11}$ is linear over $+$. Also, $\Omega_{11}$ does not affect the left-most point of $\mathcal{C}^\circ/\mathcal{C}^{\circ *}$ and replaces the right-most point of $\mathcal{C}^*/\mathcal{C}^{\circ *}$. See

Figure 3.8.

$$
\begin{aligned}
\Omega_{11}(\mathcal{Z}) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} \\
\Omega_{11}(\mathcal{Z}^*) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} \\
\Omega_{11}(\mathcal{Z}^\circ) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_{11}(\mathcal{Z}^{\circ *}) &= \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}^\circ \\
\Omega_{11}(\mathcal{T}_h) &= \begin{cases} \left.\begin{array}{l} \Omega_{11}(X_1^*)\Omega_0(X_2\cdots X_m)+ \\ +\Omega_{10}(X_1^*)\Omega_{01}(X_2\cdots X_m) \end{array}\right\} & \text{if } k=1 \\ \left.\begin{array}{l} \Omega_{11}(X_1)\Omega_0(X_2\cdots X_m)+ \\ +\Omega_{10}(X_1)\Omega_{01}(X_2\cdots X_m)+ \\ +\Omega_0(X_1)\Omega_{11}(X_2\cdots X_m) \end{array}\right\} & \text{if } k>1. \end{cases}
\end{aligned}
\tag{3.10}
$$

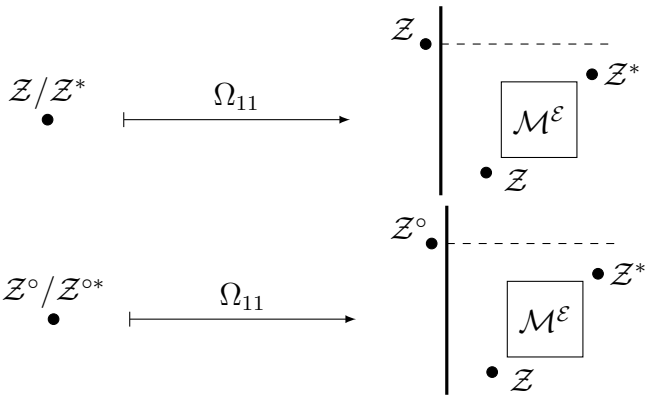



Figure 3.8: $\Omega_{11}$: juxtaposing a monotone sequence with tracked left-most and right-most most points to the right of a single point. The RHS now contains the right-most point whether we started with $\mathcal{Z}, \mathcal{Z}^*, \mathcal{Z}^\circ$ or $\mathcal{Z}^{\circ *}$. The original point was the left-most, this property of it remains unaffected. We denote $\mathcal{M}+\mathcal{E}$ by $\mathcal{M}^{\mathcal{E}}$.

To help us with enumeration, we append *phantom* points to permutations or permutation classes. By phantom, we mean a temporary extra point that is not really supposed to be there. The reason we need to do such a thing comes up in the situation portrayed in Figure 3.9. We said that in a juxtaposition we associate monotone sequences (on the RHS) with points (on the LHS) by placing them into the gaps directly below those points, see Figure 3.3. However, in a juxtaposition, the points on the RHS can occur above the top-most point on the LHS as well. So we are one gap short. To remedy this, we add a temporary point to the permutation

on the LHS. It is placed above all other points on the LHS and thereby creating the additional gap that we were missing. We then remove this extra point from our counting expression. Next, we offer a formal definition of a phantom point.

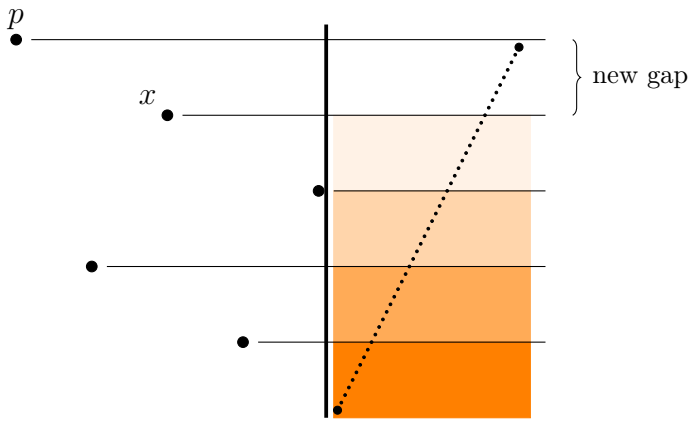



Figure 3.9: Compare with Figure 3.3. Phantom point $p$ created an extra gap where we can place a monotone sequence — above the topmost point $x$ of $P = 2413$. When done with enumeration, it is necessary to remove $p$ as it is not part of $P$ and we do not want it to distort the generating function.

**Definition 3.1.4** (Phantom point)**.** Let $P$ be a permutation from $\mathcal{C}$. An *upper phantom point* $p$ of $P$ is a point temporarily added to $P$ that has value $|P|+1$ and position 0. In other words, if $P'$ denotes $P$ with an added upper phantom point, then

$$P' = p \ominus P.$$

We sometimes refer to $P$ as an upper phantom point of $\mathcal{C}$, meaning that every $P$ in $\mathcal{C}$ is treated as $P' = p \ominus P$. A *lower phantom point* of $P$, usually denoted by $q$, is a point external to $P$ that has value 0 and position 0. In other words, if $P'$ denotes $P$ with an added lower phantom point, then

$$P' = q \oplus P.$$

We always need only one phantom point at a time and we will always specify which one. If $\mathcal{C}$ is equipped with a phantom point, it is denoted by $\overline{\mathcal{C}}$.

Given that phantom points act as usual points of a permutation, they get acted on by $\Omega$ operators. Exactly as intended. It will become clear why we need the

lower phantom point when we append decreasing sequences on the right side of context-free classes.

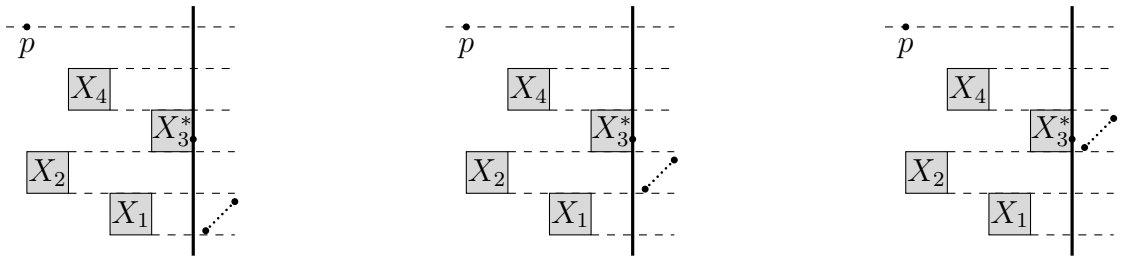
Before we give an example of how to apply $\Omega_{11}$ and how a phantom point behaves in practice, let us phrase one last observation as a lemma. It will be useful in the forthcoming proofs and examples.

**Lemma 3.1.1.** *The following operators "ignore" the right-most points (stars) in their arguments as indicated below.*

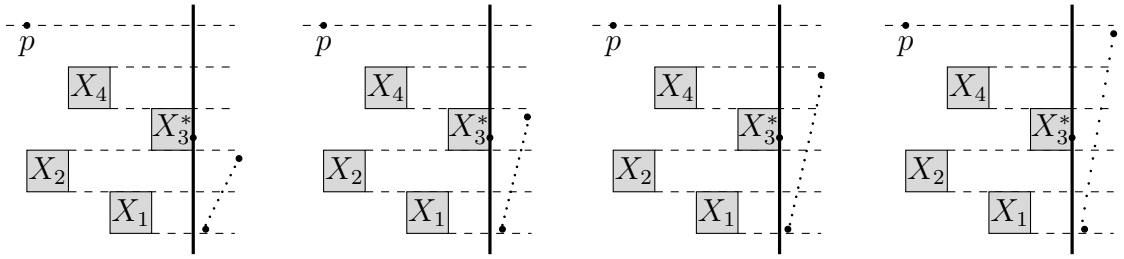
1. $\Omega_0(\mathcal{C}^*) = \Omega_0(\mathcal{C}) = \mathcal{C}$
2. $\Omega_\infty(\mathcal{C}^*) = \Omega_\infty(\mathcal{C})$
3. $\Omega_{01}(\mathcal{C}^*) = \Omega_{01}(\mathcal{C})$.

*Proof.* First, notice that any term in any polynomial from a context-free specification of some $\mathcal{C}$ remains unaffected by $\Omega_0$. This follows from the definition. Second, again by definition, $\Omega_0$ acts identically on starred and starless objects. This proves the first item. For the same reasons, action of $\Omega_\infty$ and $\Omega_{01}$ does not depend on the rightmost point in the class they take as argument. □

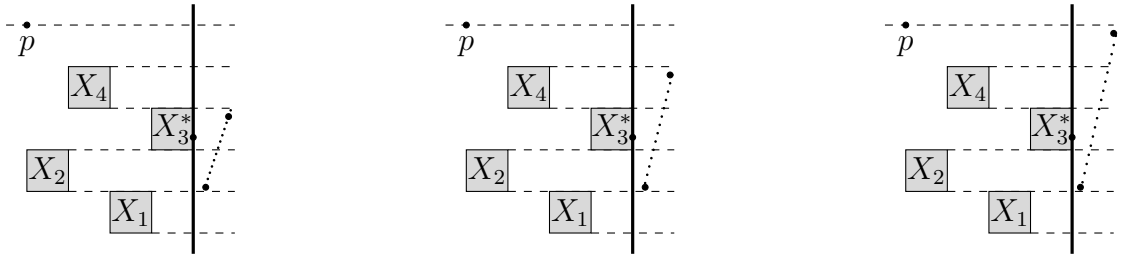
Having defined all necessary concepts, let us consider an example of $\Omega_{11}$ acting on a term $\mathcal{T}_h = X_1X_2X_3^*X_4p = X_1X_2X_3^*X_4\mathcal{Z}$, where $p$ is an upper phantom point. For no particular reason we assume that $X_1X_2X_3X_4$ are classes inflating 2413. See Figure 3.10 for pictorial representation of the juxtapositions as well as their descriptions in terms of $\Omega$ operators in the captions. We use Lemma 3.1.1 to simplify the expressions by omitting $\Omega_0$ operator, e.g. $\Omega_0(\mathcal{Z})$ reduces to $\mathcal{Z}$. The Figure 3.10a shows the terms in the recursion which only rely on $\Omega_{11}$ and $\Omega_0$ operators. The remaining terms do not use $\Omega_{11}$, however they additionally employ $\Omega_\infty$, $\Omega_{10}$, and $\Omega_{01}$ operators.




(a) Left to right: $\Omega_{11}(X_1)X_2X_3X_4\mathcal{Z}$, $X_1\Omega_{11}(X_2)X_3X_4\mathcal{Z}$, and $X_1X_2\Omega_{11}(X_3^*)X_4\mathcal{Z}$.




(b) Left to right: $\Omega_{10}(X_1)\Omega_{01}(X_2)X_3X_4\mathcal{Z}$, $\Omega_{10}(X_1)\Omega_{\infty}(X_2)\Omega_{01}(X_3^*)X_4\mathcal{Z}$, $\Omega_{10}(X_1)\Omega_{\infty}(X_2X_3^*)\Omega_{01}(X_4)\mathcal{Z}$, and $\Omega_{10}(X_1)\Omega_{\infty}(X_2X_3^*X_4)\Omega_{01}(\mathcal{Z})$.




(c) Left to right: $X_1\Omega_{10}(X_2)\Omega_{01}(X_3^*)X_4\mathcal{Z}$, $X_1\Omega_{10}(X_2)\Omega_{\infty}(X_3^*)\Omega_{01}(X_4)\mathcal{Z}$, and $X_1\Omega_{10}(X_2)\Omega_{\infty}(X_3^*X_4)\Omega_{01}(\mathcal{Z})$.


(d) Left to right: $X_1X_2\Omega_{10}(X_3^*)\Omega_{01}(X_4)\mathcal{Z}$ and $X_1X_2\Omega_{10}(X_3^*)\Omega_{\infty}(X_4)\Omega_{01}(\mathcal{Z})$.

Figure 3.10: We apply $\Omega_{11}$ to $\mathcal{T} = X_1X_2X_3^*X_4p$. Inflation $2413[X_2, X_4, X_1, X_3]$ represents $X_1X_2X_3^*X_4$. Lowest and top-most points on the RHS are bold (distinguished under $\Omega_{11}$). Lowest point on RHS is below right-most point on LHS.

## 3.2 Main results

Just briefly in this first paragraph we use context-free to refer to "context-free admitting a combinatorial specification tracking the right-most point". This section presents results that we are able to obtain with the tools set up in Section 3.1. First, we prove that a juxtaposition of a non-empty context-free cell with a non-empty monotone increasing cell is context free. This invariant is stated as Lemma 3.2.1 and we need it for the proof of the key result, Proposition 3.2.1. In Proposition 3.2.1, we state that appending monotone increasing classes on the right side of a context-free permutation class does not change the character of the original combinatorial specification and hence the generating function of the obtained juxtaposition class is algebraic. Next we prove that by left-right flips and up-down flips, we can rephrase appending a decreasing monotone permutation class into appending an increasing monotone permutation class, and we can do so on either side of $\mathcal{C}$ (left or right). This is established in Propositions 3.2.2 and 3.2.3. Therefore, it turns out that appending a monotone class (increasing or decreasing) on either side of a context-free permutation class preserves the character of the combinatorial specification and hence the character of the generating function of such a class. We use induction to prove that after finitely many such juxtapositions, the resulting class is still context-free and is enumerated by an algebraic generating function. This is our main result and is stated as Theorem 3.2.1. In addition to this general setting, we show in a Corollary 3.2.1 that the same holds for all context-free permutation classes with finitely many simple permutations. These form a large special case.

Let us now state the key lemma expressing the "context-freeness invariant" with care.

**Lemma 3.2.1.** *Let $\mathcal{C}$ be a context-free permutation class and let $\mathscr{S}$ be its combinatorial specification which tracks the rightmost point of $\mathcal{C}$ by vertical position (value). Let $\mathcal{M}$ be a monotone increasing permutation class. Consider a class $\mathcal{C}'$ of permutations $P$ which admit our usual leftmost gridding as $P = P_1|P_2$ where $P_1 \in \mathcal{C}$ and $P_2 \in \mathcal{M}$ and neither $P_1$ nor $P_2$ is empty. Then $\mathcal{C}'$ is context-free and admits a combinatorial specification $\mathscr{S}'$ that tracks the rightmost point by vertical position (value).*

*Proof.* In the language of $\Omega$ operators, expressing $\mathcal{C}'$ requires installing a phantom point $p$ above $\mathcal{C}$ to construct $\overline{\mathcal{C}}^*$. Furthermore, we will express $\mathcal{C}'\mathcal{Z}$ instead of $\mathcal{C}'$ as removing the phantom point from each permutation in $\mathcal{C}'\mathcal{Z}$ is simply a matter of removing that trailing $\mathcal{Z}$ from every term (product) $\mathcal{T}$ in its combinatorial specification. Notice that every term $\mathcal{T}$ indeed has such $\mathcal{Z}$ at the end of it as every term has a phantom point on top. This means that if $\mathcal{C}'\mathcal{Z}$ has a context-free specification that tracks the rightmost point by value, then so does $\mathcal{C}'$. Therefore, the expression for $\mathcal{C}'\mathcal{Z}$ is the following one.

$$\mathcal{C}'\mathcal{Z} = \Omega_1(\overline{\mathcal{C}}^*) + \Omega_{11}(\overline{\mathcal{C}}^*), \tag{3.11}$$

It now remains to show two things

1. $\Omega_1(\overline{\mathcal{C}}^*) + \Omega_{11}(\overline{\mathcal{C}}^*)$ indeed represents $\mathcal{C}'\mathcal{Z}$.
2. $\Omega_1(\overline{\mathcal{C}}^*)$ and $\Omega_{11}(\overline{\mathcal{C}}^*)$ admit context-free combinatorial specifications that track the rightmost points.

If 1. is true, then we simply know that we are proving the right thing. If also 2. is true, then $\Omega_1(\overline{\mathcal{C}}^*) + \Omega_{11}(\overline{\mathcal{C}}^*)$ also admits a context-free combinatorial specification that tracks the rightmost point and by 1. we have indeed shown that the conclusion of Lemma 3.2.1 holds.

Let us begin by proving 1. This follows from the definitions of $\Omega_1$ and $\Omega_{11}$. We only briefly describe how as it is straightforward yet tedious. One has to accept that $\Omega_1$ represents a cell with a single point juxtaposed on the right of it so that this point is always below the rightmost point of the left cell (to ensure the leftmost possible gridding). We recall the definition of $\Omega_1$.

$$\begin{aligned}
\Omega_1(\mathcal{Z}) &= \mathcal{Z}^*\mathcal{Z} \\
\Omega_1(\mathcal{Z}^*) &= \mathcal{Z}^*\mathcal{Z} \\
\Omega_1(\mathcal{T}_h) &= \begin{cases} \Omega_1(X_1^*)\Omega_0(X_2\cdots X_m) & \text{if } k = 1 \\ \Omega_1(X_1)\Omega_0(X_2\cdots X_m) + \Omega_0(X_1)\Omega_1(X_2\cdots X_m), & \text{if } k > 1. \end{cases}
\end{aligned}$$

The base cases clearly do juxtapose a point on the RHS below the point on the LHS. See Figure 3.5 and recall that reading the Cartesian product left-to-right reflects

the bottom-to-top order in the permutation. As for the recursive step, notice that if the rightmost point is in the bottom-most term of the product, then we do not apply $\Omega_1$ above it at all (and so the leftmost point on the RHS is not above the rightmost point on the LHS). Operator $\Omega_0$ is applied to the rest of the terms in the product, meaning that there is nothing on the RHS next to those terms. If the bottom-most term of the product does not contain the rightmost point, then there are two options. Either the single point on the RHS is next to this term or it is not. These are the two expressions on the second line of the recursive step. Notice that second term involves a recursion on the tail of the product which is strictly shorter in length than $\mathcal{T}_h$, so the procedure terminates. And therefore, $\Omega_1$ indeed represents a juxtapositions of two non-empty cells as desired, provided that its argument is non-empty (which we make sure it is). To show that $\Omega_{11}$ correctly expresses the juxtaposition we want is slightly more complicated. The definition of $\Omega_{11}$ depends on the definitions of $\Omega_{10}, \Omega_{01}, \Omega_\infty$ and requires a more tedious verification which we do not do here. However, in Figure 3.10 we demonstrate the definition of $\Omega_{11}$ on a simple $\mathcal{T}_h$ to make it easier to work through the definition. We conclude that point 1. is valid.

To show that 2. holds, we need to revisit the definitions of $\Omega_1$ and $\Omega_{11}$ again. We work our way through the definition $\Omega_1$ and leave $\Omega_{11}$ to the reader as it is analogous but significantly longer. The goal is to show that $\Omega_1$ preserves the context-freeness of the combinatorial specification of its input and correctly replaces the right-most point in the input with the new right-most point (the single point on the RHS). Looking at the definition above, $\Omega_1$ does two things. One, it replaces $\mathcal{Z}$ with $\mathcal{Z}^*\mathcal{Z}$ or $\mathcal{Z}^*$ with $\mathcal{Z}^*\mathcal{Z}$. Two, it splits a product $\mathcal{T}_h$ into two products: $\Omega_1(X_1)\Omega_0(X_2 \cdots X_m)$ and $\Omega_0(X_1)\Omega_1(X_2 \cdots X_m)$. Each of these products has $\Omega_1$ applied to a smaller expression than $\mathcal{T}_h$ and by induction those are of the correct form. The rest of each of the two products has $\Omega_0$ applied to it which does not change that part of the expression. Therefore, given that $\Omega_1$ is linear over $+$, it indeed preserves the context-freeness of the combinatorial specification of its input. We argued that the rightmost point gets tracked correctly in 1. This concludes our task for $\Omega_1$. Similarly, one can verify that $\Omega_{11}$ produces a context-free output that tracks the rightmost point, provided the input is of this form.

As shown before, 1. together with 2. imply the statement of Lemma 3.2.1. □

**Proposition 3.2.1.** *Let $\mathcal{C}$ be a context-free permutation class and $\mathscr{S}$ its combinatorial specification which tracks the rightmost point of $\mathcal{C}$ by vertical position (value). Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be a sequence of monotone increasing permutation classes. Then $\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_k$ admits a context-free combinatorial specification which tracks the rightmost point by value. Consequently, $\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_k$ admits an algebraic generating function.*

*Proof.* We prove by induction that for every $k \geq 1$, the class $\mathcal{C}|\mathcal{M}_1|\ldots|M_k$ admits a context-free combinatorial specification which tracks the rightmost point by value. It then follows that there also exists a context-free combinatorial specification of such a class, simply by dropping $^*$ and $^\circ$ decorations. Consequently, it follows by Theorem 3.1.1 that $\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_k$ admits a generating function that is algebraic.

First, notice that we can rewrite the original juxtaposition as follows to obtain disjoint terms and every cell in each term non-empty.

$$\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_k = \mathcal{E} + \mathcal{M}'_k + \ldots + \mathcal{M}'_1|\ldots|\mathcal{M}'_k + \mathcal{C}'|\mathcal{M}'_1|\ldots|\mathcal{M}'_k, \tag{3.12}$$

where, for the moment, $\mathcal{M}'$ and $\mathcal{C}'$ denote the non-empty versions of the respective classes. To see where each term comes from, recall that we grid greedily from the right. So if all $k$ cells are empty, then the cell with $\mathcal{C}$ must also be empty and this is represented by $\mathcal{E}$. If there is one non-empty cell, then it must be $\mathcal{M}_k$ — again thanks to the greedy gridding from the right. We continue this way and obtain exactly (3.12). Furthermore, since all $\mathcal{M}_i$ are identical, it follows that $\mathcal{M}'_1|\ldots|\mathcal{M}'_{k-1} = \mathcal{M}'_2|\ldots|\mathcal{M}'_k$. We will use this during induction.

**Base case:**
If $k = 1$, then (3.12) has the following form.

$$\mathcal{C}|\mathcal{M}_1 = \mathcal{E} + \mathcal{M}'_1 + \mathcal{C}'|\mathcal{M}'_1 \tag{3.13}$$

Clearly, $\mathcal{M}'_1$ is context-free and admits a combinatorial specification which tracks the rightmost point by value (we saw this in Section 3.1 as $\mathcal{M}^*$). By Lemma 3.2.1, $\mathcal{C}'|\mathcal{M}'_1$ is also context-free and admits a combinatorial specification which tracks

the rightmost point by value. Therefore, $\mathcal{C}|\mathcal{M}_1$ is context-free and admits the right combinatorial specification — we are done.

**Induction step:**
For any $k$ greater than 1, let $\mathcal{C}'_0 = \mathcal{C}'|\mathcal{M}'_1|\ldots|\mathcal{M}'_{k-1}$. By induction assumption, $\mathcal{C}'_0$ is context-free and admits a combinatorial specification that tracks the rightmost point by value. This is the case as $\mathcal{C}_0 = \mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_{k-1}$ is all that by induction assumption and $\mathcal{C}'_0$ is one term of an expression like (3.12) for $\mathcal{C}_0$. Therefore, we need to show that $\mathcal{C}'_0|\mathcal{M}'_k$ is context-free and admits the right combinatorial specification. But $\mathcal{C}'_0$ and $\mathcal{M}'_k$ meet the conditions of Lemma 3.2.1 and hence the claim follows. The only term in (3.12) which does not fall within the induction assumption in an obvious way is $\mathcal{M}'_2|\ldots|\mathcal{M}'_k$. There was no $1 \times k-1$ row of non-empty monotone cells in an expression like (3.12) for $\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_{k-1}$. On the other hand, $\mathcal{M}_2$ is a context-free class that admits a combinatorial specification which tracks the rightmost point by value. Therefore, let $\mathcal{C}_{\mathcal{M}} = \mathcal{M}_2$, and we get $\mathcal{M}'_2|\ldots|\mathcal{M}'_k = \mathcal{C}'_{\mathcal{M}}|\mathcal{M}'_3|\ldots|\mathcal{M}'_k$. Now, this juxtaposition of non-empty cells does fall under the induction assumption and we are done. Every term in (3.12) is context-free and admits a combinatorial specification which tracks the rightmost point by value. Therefore, $\mathcal{C}|\mathcal{M}_1|\ldots|\mathcal{M}_k$ is such a context-free class and the Proposition 3.2.1 follows. □

The major message in Lemma 3.2.1 is that juxtaposing a monotone increasing class on the right of a context-free class does not qualitatively affect the context-free class that we started with. This turns out to be a key invariant as it lends itself to repeated juxtapositions. That is the other crucial message and we introduced it in Proposition 3.2.1 as "repeated juxtaposing is allowed" for monotone increasing classes on the right side of a context-free class. These two building blocks are going to be used as black boxes when we generalise the increasing monotone classes to any monotone classes and allow the juxtapositions on both sides of the context-free class.

### 3.2.1 Extension to decreasing classes and both sides

As advertised, this subsection transforms more general juxtapositions into a form that Lemma 3.2.1 and Proposition 3.2.1 can deal with. The culmination is Theorem 3.2.1 which concludes that finitely many juxtapositions of any monotone classes on any side of a context-free class form a context-free class, which thereby admits an algebraic generating function.

We begin by proving that it is possible to append decrasing monotone classes on the right side of a context-free class.

**Proposition 3.2.2.** *Let $\mathcal{D}$ be a monotone decreasing permutation class and $\mathcal{C}$ a context-free permutation class that admits a combinatorial specification which tracks the right-most point by value. Then $\mathcal{C}|\mathcal{D}$ admits a combinatorial specification which tracks the rightmost point by value.*

*Proof.* We reduce the juxtaposition $\mathcal{C}|\mathcal{D}$ to a juxtaposition of the form $\mathcal{J}|\mathcal{M}$ (an upside-down $\mathcal{C}|\mathcal{D}$). However, there are several details that require attention.For instance, in a juxtaposition $\mathcal{C}|\mathcal{M}$, a point $x$ on the LHS is associated with a sequence on the RHS that falls into the gap immediately below $x$, see Figure 3.3. If we want this to be the case for the juxtaposition $\mathcal{J}|\mathcal{M}$, we must start by associating points on the RHS of $\mathcal{C}|\mathcal{D}$ *above* the points on the LHS. Similarly, If we want an upper phantom point $p$ to be above and to the left of $\mathcal{J}|\mathcal{M}$, we must start with a lower phantom point $q$ as in $q \oplus \mathcal{C}|\mathcal{D}$. Given this set-up, we write $\mathcal{C}|\mathcal{D}$ as a sum of disjoint terms.

$$\mathcal{C}|\mathcal{D} = \mathcal{E} + \mathcal{D}' + \mathcal{C}'|\mathcal{D}', \tag{3.14}$$

where $\mathcal{C}'$ and $\mathcal{D}'$ are non-empty versions of $\mathcal{C}$ and $\mathcal{D}'$, respectively.

We transform $q \oplus \mathcal{C}^*|\mathcal{D}$ into $p \ominus \mathcal{J}^*|\mathcal{M}$ by a flip along the horizontal axis — otherwise known as complementation. We call the operator that does this job $\Theta$. The rightmost point stays right-most after the flip, i.e. $\Theta(^*) = {}^*$. The phantom point $q$ assumes the usual position of the phantom point $p$, i.e. $\Theta(q) = p$, a mere renaming. The points on the RHS now appear below the ones on the LHS that they are associated with. It goes without saying that a bottom-to-top specification can be simply reversed into a top-to-bottom specification in order to facilitate the

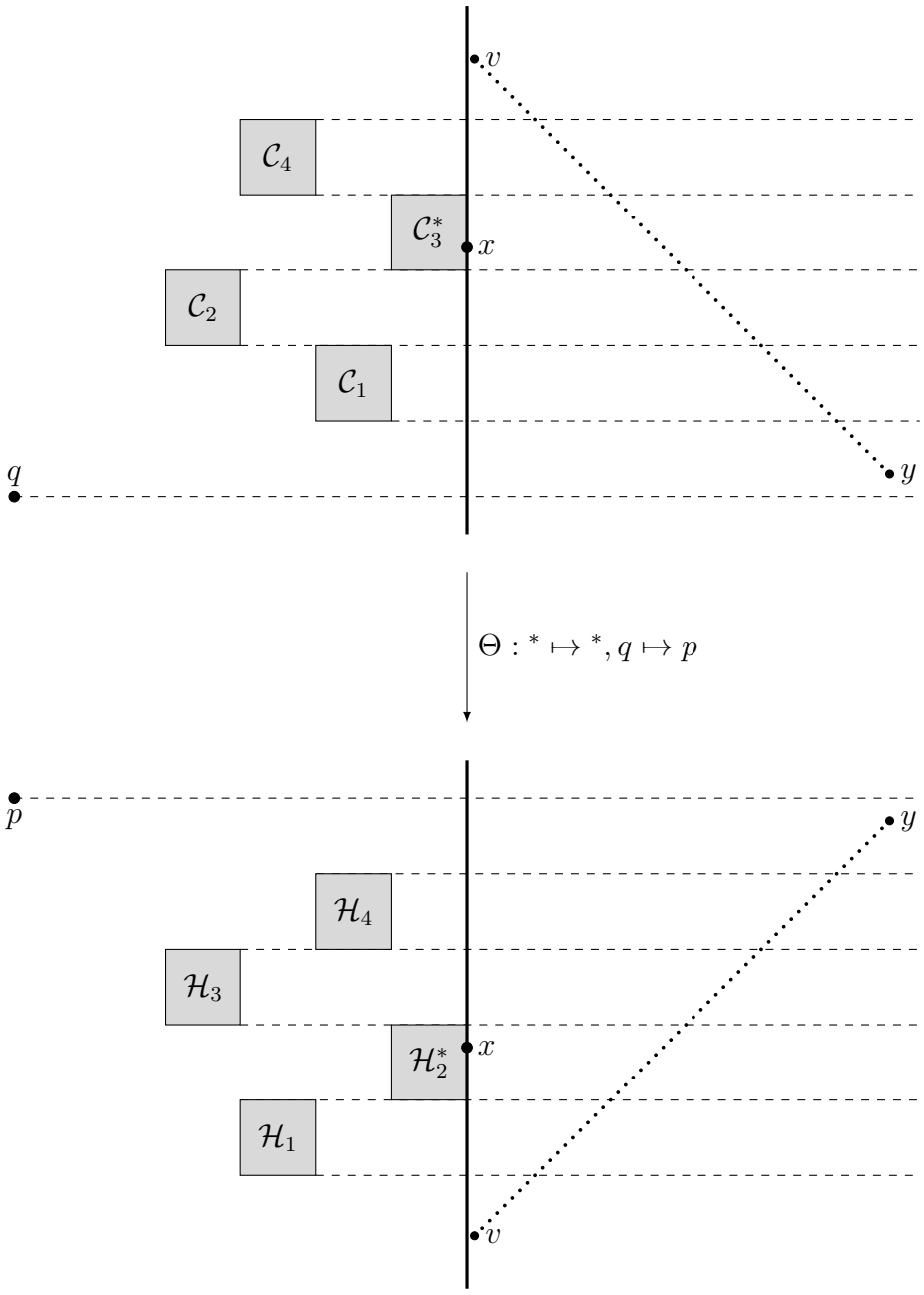



Figure 3.11: Appending a monotone decreasing class to $\mathcal{C}$ amounts to appending a monotone increasing class to $\mathcal{C}$ upside-down. In the notation of the figure, $\mathcal{C}_1 = \mathcal{H}_4$, $\mathcal{C}_2 = \mathcal{H}_3$, $\mathcal{C}_3^* = \mathcal{H}_2^*$, and $\mathcal{C}_4 = \mathcal{H}_1$. Before the flip one needs to associate decorations with gaps above LHS points, not below LHS points. Additionally, the lower phantom point which is transformed to the upper phantom point by the upside-down flip.

upside-down flip. The entire situation is analogous to the one in Lemma 3.2.1. The only term we need to worry about is when both $\mathcal{C}$ and $\mathcal{D}$ are non-empty. Writing the juxtpaosition of two non-empty cells in terms of $\Omega$ operators gives the following expression.

$$\mathcal{C}'|\mathcal{D}' = \Theta^{-1}(\Omega_1(\Theta(q \oplus \mathcal{C}^*))) + \Theta^{-1}(\Omega_{11}(\Theta(q \oplus \mathcal{C}^*))).$$

In other words, we can use all the infrastructure that is in place for monotone increasing classes to append monotone decreasing classes. Since $\Theta$ is bijective, we transform the decreasing setting into increasing, apply the operators we need to apply, and then bring the situation back by $\Theta^{-1}$. Figure 3.11 gives an instance of the upside-down flip transformation. □

**Proposition 3.2.3.** *Let $\mathcal{D}$ be a monotone decreasing permutation class and $\mathcal{C}$ a context-free permutation class that admits a combinatorial specification which tracks the leftmost point. Then $\mathcal{D}|\mathcal{C}$ is context-free and admits a combinatorial specification which tracks the leftmost.*

*Proof.* Similar to the upside-down flip in Proposition 3.2.2, the proof proceeds by transformation of $\mathcal{C}$ via a left-to-right flip $\Phi$, then applying $\Omega$ operators, and undoing the flip. In the process, we need to make sure that the left-most and the rightmost points are treated correctly, and that the phantom points work as expected. For this purpose, we need to keep track of the left-most point of $\mathcal{C}$. First, we require that the combinatorial specification of $\mathcal{C}$ tracks the leftmost point of $\mathcal{C}$. Recall that we denote the objects containing the leftmost point by $\mathcal{C}^\circ$ or $\mathcal{Z}^\circ$. As before, enumerating $\mathcal{D}|\mathcal{C}$ amounts to enumerating the juxtaposition of two non-empty cells where the one on the left-hand side is $\mathcal{D}'$ and the one on the right-hand side is $\mathcal{C}'$. If $\Phi$ is the left-to-right flip operator, then enumerating $\mathcal{D}|\mathcal{C}$ is the same as enumerating $\Im|\mathcal{M}$, where $\mathcal{M}$ stands for monotone increasing class, if we remember to keep track of the details (critical points). In particular, $\Phi(\mathcal{Z}^\circ) = \Phi(\mathcal{Z}^*)$ and $\Phi(\mathcal{Z}^*) = \Phi(\mathcal{Z}^\circ)$. In contrast to the upside-down flip operator $\Theta$, we begin with appending an invalid phantom point to $\mathcal{C}$, which becomes valid after the flip, and is removed afterwards when $\Phi$ is undone. Please refer to Figure 3.12 for a visual

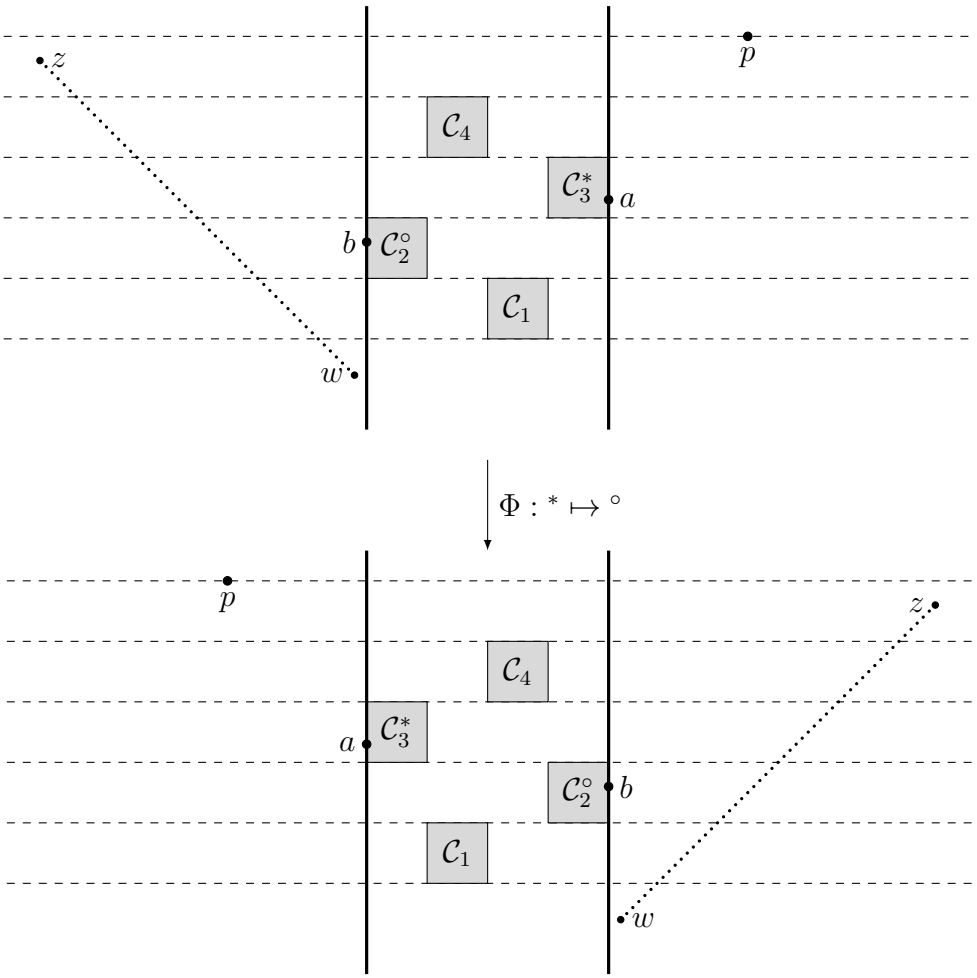



Figure 3.12: Left-to-right flip swaps starred and circled points (or classes) and causes the monotone class to change from decreasing to increasing or vice versa. We also begin with a point $p$ which is not a phantom point as it is to the right of $\mathcal{C}$. It becomes a phantom point after $\Phi$ is applied to the set-up. The bottom picture is the arrangement representing the action of $\Omega$ operators.

guide.

$$\mathcal{D}'|\mathcal{C}' = \Phi^{-1}(\Omega_1(\Phi(C^\circ \oplus p))) + \Phi^{-1}(\Omega_{11}(\Phi(C^\circ \oplus p))).$$

It is clear that after the transformation by $\Phi$, the arrangement is suitable for application of $\Omega$ operators. The correctness follows from Lemma 3.2.1 and Proposition 3.2.1. Therefore, $\mathcal{D}|\mathcal{C}$ admits a combinatorial specification which tracks the rightmost point. □

Notice that by Proposition 3.2.2 and Proposition 3.2.3, appending increasing classes on the left-hand side also becomes possible. We wrap up our efforts in the following theorem, Theorem 3.2.1.

**Theorem 3.2.1.** *Let $\mathcal{C}$ be a context-free permutation class that admits a combinatorial specification which tracks both the right-most and the left-most points. Let $\mathcal{M}_1, \ldots, \mathcal{M}_{k+\ell}$ be a sequence of monotone, increasing or decreasing, permutation classes. Then $\mathcal{M}_1|\ldots|\mathcal{M}_k|\mathcal{C}|\mathcal{M}_{k+1}|\ldots|\mathcal{M}_{k+\ell}$ is a context-free permutation class that admits an algebraic generating function.*

*Proof.* Once we decide which gridding (from the left side or from the right side) gets priority, the claim follows by repeated applications of Proposition 3.2.1, Proposition 3.2.2, and Proposition 3.2.3. Recall that the $\Omega$ operators only use the right-most point and ignore the presence of the left-most point (not deleting the circle, simply leaving it unchanged). Having both decorations ($^*$ and $^\circ$) present in $\mathcal{C}$ at the same time does not negatively affect nor gets negatively affected by $\Omega$ operators.

In short, Lemma 3.2.1 together with Proposition 3.2.1 guarantee that whatever we start with before the next juxtaposition is context-free and tracks the desired points correctly. Additionally, Proposition 3.2.1 also deals with the case when we append an increasing class on the right. The remaining cases are covered by first manipulating the situation via Proposition 3.2.2 and Proposition 3.2.3 into the arrangement when Proposition 3.2.1 applies again. Therefore, juxtaposing finitely many monotone classes on both sides of a context-free $\mathcal{C}$ leads to just another context-free class. Therefore, Theorem 3.1.1 then implies that the generating function of the resulting class is in fact algebraic. □

**Corollary 3.2.1.** *Let $\mathcal{C}$ be a context-free permutation class with finitely many simple permutations and let $\mathcal{M}_1, \ldots, \mathcal{M}_{k+\ell}$ be as in Theorem 3.2.1. Then*

$$\mathcal{M}_1|\ldots|\mathcal{M}_k|\mathcal{C}|\mathcal{M}_{k+1}|\ldots|\mathcal{M}_{k+\ell}$$

*is a context-free permutation class which admits a generating function that is an algebraic function.*

*Proof.* We show that every context-free permutation class with finitely many simple permutations admits a combinatorial specification that tracks the right-most

and left-most points. The claim then follows from Theorem 3.2.1.

If $\mathrm{Si}(\mathcal{C})$ is the set of simple permutations in $\mathcal{C}$, we say that a vector $(\mathcal{C}_1, \ldots, \mathcal{C}_{|\sigma|})$ is admissible, if $\sigma \in \mathrm{Si}(\mathcal{C})$ and $\sigma[\alpha_1, \ldots, \alpha_{|\sigma|}]$ is in $\mathcal{C}$ for all choices of $\alpha_i \in \mathcal{C}_i$. In case of 12 and 21, we insist that $\mathcal{C}_1$ in an admissible vector $(\mathcal{C}_1, \mathcal{C}_2)$ is always sum-indecomposable and skew-indecomposable, respectively. For convenience, we temporarily violate the customary notation about inflations. We use $\mathcal{C}_i$ to denote the $i$-th class from the bottom in the inflation of $\sigma$, i.e. $\sigma[C_1, \ldots, \mathcal{C}_i, \ldots, \mathcal{C}_{|\sigma|}]$ means that the bottom-most point in $\sigma$ is inflated by $\mathcal{C}_1$, the point immediately above it in $\sigma$ is inflated by $\mathcal{C}_2$, and so on. In other words, we list the inflation by value instead of by position. Additionally, assume that the point with value $k_\sigma$ in $\sigma$ is the rightmost point of $\sigma$ and the point with value $\ell_\sigma$ is the left-most point of $\sigma$. Furthermore, we use a variable $X$ to stand for a class from $\{\mathcal{C}_1, \ldots, \mathcal{C}_r\}$. We now give the combinatorial specification template that every context-free class $\mathcal{C}$ with finitely many simples fits into. We only define the classes with both the left-most and the right-most points tracked. However, the combinatorial specification needs definitions of starred, circled, starles and circleless classes as well. Their definitions are analogous and take up a lot of space. We skip them.

$$\begin{aligned}
\mathcal{C}_1^{\circ *} &= \mathcal{Z} + \sum_{\substack{\text{admissible} \\ (X_1, X_2)}} 12[X_1^\circ, X_2^*] + \sum_{\substack{\text{admissible} \\ (X_1, X_2)}} 21[X_1^*, X_2^\circ] + \\
&\quad + \sum_{\sigma \in \mathrm{Si}(\mathcal{C})} \sum_{\substack{\text{admissible} \\ (\mathcal{C}_1, \ldots, \mathcal{C}_k^*, \ldots, \mathcal{C}_\ell^\circ, \ldots, \mathcal{C}_{|\sigma|})}} \sigma[\mathcal{C}_1, \ldots, \mathcal{C}_k^*, \ldots, \mathcal{C}_\ell^\circ, \ldots, \mathcal{C}_{|\sigma|}] \\
&\quad \vdots \\
\mathcal{C}_r^{\circ *} &= \mathcal{Z} + \sum_{\substack{\text{admissible} \\ (X_1, X_2)}} 12[X_1^\circ, X_2^*] + \sum_{\substack{\text{admissible} \\ (X_1, X_2)}} 21[X_1^*, X_2^\circ] + \\
&\quad + \sum_{\sigma \in \mathrm{Si}(\mathcal{C})} \sum_{\substack{\text{admissible} \\ (\mathcal{C}_1, \ldots, \mathcal{C}_k^*, \ldots, \mathcal{C}_\ell^\circ, \ldots, \mathcal{C}_{|\sigma|})}} \sigma[\mathcal{C}_1, \ldots, \mathcal{C}_k^*, \ldots, \mathcal{C}_\ell^\circ, \ldots, \mathcal{C}_{|\sigma|}]
\end{aligned} \tag{3.15}$$

This completes the proof. □

Section 3.3.3 gives an example of a very simple class with finitely many simple permutations — the class of separable permutations, or $\mathcal{S}$. We know $\mathrm{Si}(\mathcal{S}) = \emptyset$

and therefore, the last terms on the RHS in (3.15) vanishes. We enumerate exactly the juxtaposition of the class of separable permutations with the class of monotone increasing permutations.

Notice that Corollary 3.2.1 is strictly weaker than Theorem 3.2.1. For instance, Av(321) is a class with infinitely many simple permutations. Indeed, 2413, 246135, 24681357, ... are all 321-avoiders and all of them are simple. Yet Av(321) admits a combinatorial specification required by Theorem 3.2.1. We show this in Section 3.3.1 where we juxtapose it with a monotone increasing class and enumerate this juxtaposition exactly using methods developed in Section 3.1.

In Section 3.3.2, we enumerate exactly a repeated juxtaposition of the form $\mathcal{C}|\mathcal{M}|\mathcal{M}$ where $\mathcal{C} = \mathcal{M}$ and $\mathcal{M}$ is a monotone increasing class.

## 3.3 Applications to exact enumeration

### 3.3.1 Example: Av(321|21)

In Chapter 2 we deal with the juxtaposition of a Catalan class with a monotone class by enumerating all such juxtapositions. Here, we re-enumerate one of these cases, which also serves to illustrate that the method applies to classes other than those with finitely many simples. This demonstrates the gap between the Corollary 3.2.1 and Theorem 3.2.1.

We represent permutations in Av(321) by Dyck paths below the diagonal, starting in the bottom left corner and going to the top right corner of the grid. An exact definition of Dyck path can be found in Section 2.2, including Figure 2.2. Also, below we replicate Figure 2.4 that demonstrates how we associate permutations from Av(321) with Dyck paths.

Having set the scene, we can start the enumeration. Let $\mathcal{C} := \text{Av}(321)$. In a Dyck path, we denote the right step with $R$ and the up step with $U$. We choose the following combinatorial specification that tracks the right-most point of $\mathcal{C}$ (we do not need to track the left-most point).

$$\mathcal{C}^* = (\mathcal{C} + \mathcal{E})R\mathcal{C}^*U + (\mathcal{C} + \mathcal{E})RU^*$$

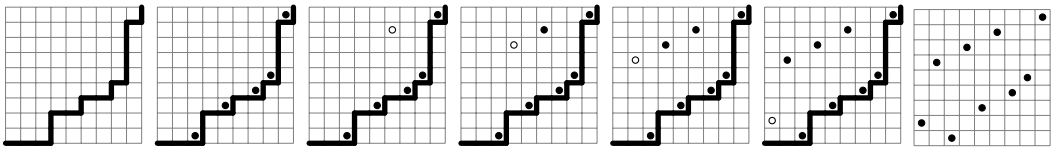


Figure 3.13: An example of a reversible process (left to right) of associating a unique 321-avoider with a given Dyck path.

$$\mathcal{C} = (\mathcal{C} + \mathcal{E})R(\mathcal{C} + \mathcal{E})U.$$

In a Dyck path, $U$ and $R$ are like brackets – they are interpreted in pairs. Hence, we rewrite the specification into the canonical form (3.16).

$$\begin{aligned} \mathcal{C}^* &= \mathcal{C}\mathcal{C}^*\mathcal{Z} + \mathcal{C}^*\mathcal{Z} + \mathcal{C}\mathcal{Z}^* + \mathcal{Z}^* \\ \mathcal{C} &= \mathcal{C}\mathcal{C}\mathcal{Z} + 2\mathcal{C}\mathcal{Z} + \mathcal{Z}. \end{aligned} \tag{3.16}$$

We use this example to demonstrate the method of $\Omega$ operators in full detail. In practice, one would be able to make the analysis that follows significantly shorter by exploiting certain ad-hoc properties of this particular juxtaposition. However, we follow the recipe.

We need to apply $\Omega$ operators to $\mathcal{C}^*$ in such a way that the expression represents the juxtaposition $\mathcal{C}|\mathcal{M}$, where $\mathcal{M} = \mathrm{Av}(21)$. The *final expression* that represents $\mathrm{Av}(321|21)$ is

$$\mathcal{F} = \mathcal{E} + \mathcal{M} + \frac{\Omega_1(\overline{\mathcal{C}}^*) + \Omega_{11}(\overline{\mathcal{C}}^*)}{\mathcal{Z}} \tag{3.17}$$

Indeed, the juxtaposition is either empty (term $\mathcal{E}$ in $\mathcal{F}$), or the left cell is empty (term $\mathcal{M}$ in $\mathcal{F}$) or both cells are non-empty (terms $\Omega_1(\overline{\mathcal{C}}^*)$ and $\Omega_{11}(\overline{\mathcal{C}}^*)$ in $\mathcal{F}$ with the former representing a single point in the right cell and the latter representing a sequence of length at least two in the right cell). Notice that the situation where only the right cell is empty cannot occur because the gridline is as far left as possible. Enumeration of $\mathcal{M}$ is trivial and so the main issue will be to enumerate the last two terms of $\mathcal{F}$. The last thing to explain about (3.17) is the division by $\mathcal{Z}$ on the RHS. In short, $\mathcal{Z}^{-1}$ removes the phantom points after they are not needed

anymore. The enumeration of Av(321|21) proceeds by obtaining the generating function for $\Omega_1(\overline{\mathcal{C}}^*)+\Omega_{11}(\overline{\mathcal{C}}^*)$, correcting it by $\mathcal{Z}^{-1}$ to remove phantom points, and then adding to it the generating function for $\mathcal{E}+\mathcal{M}$. Notice that, in general, any "algebraic" post-correction does not affect the character of the generating function (if it was algebraic to begin with).

Let us first recall the specification of $\overline{\mathcal{C}}^*$.

$$\overline{\mathcal{C}}^* = \mathcal{Z} \ominus \mathcal{C}^* = \mathcal{C}^*\mathcal{Z}$$

Let $\mathcal{B}$ be the set of all the classes that need to be defined within the combinatorial specification of $\Omega_{11}(\overline{\mathcal{C}}^*)$. We start with all the terms in $\mathcal{F}$ inside $\mathcal{B}$, as we need them all to be in the combinatorial specification.

$$\mathcal{B} = \{\mathcal{E}, \mathcal{M}, \Omega_1(\overline{\mathcal{C}}^*), \Omega_{11}(\overline{\mathcal{C}}^*)\}$$

We pop $\mathcal{E}$ first, as it is trivial to define.

$$\mathcal{E} = 1 \tag{3.18}$$

Next, we pop $\mathcal{M}$ and define it below.

$$\mathcal{M} = \mathcal{Z} + \mathcal{M}\mathcal{Z} \tag{3.19}$$

The new $\mathcal{B}$ is as follows

$$\mathcal{B} = \{\Omega_1(\overline{\mathcal{C}}^*), \Omega_{11}(\overline{\mathcal{C}}^*)\}.$$

Next we pop $\Omega_1(\overline{\mathcal{C}}^*)$ and define it.

$$\begin{aligned} \Omega_1(\overline{\mathcal{C}}^*) &= \Omega_1(\mathcal{C}^*\mathcal{Z}) \\ &= \Omega_1(\mathcal{C}^*)\mathcal{Z} \end{aligned} \tag{3.20}$$

We update $\mathcal{B}$ to include the new undefined class (in red).

$$\mathcal{B} = \{\Omega_1(\mathcal{C}^*), \Omega_{11}(\overline{\mathcal{C}}^*)\}.$$

We pop the new element from $\mathcal{B}$ and define it next.

$$\begin{aligned}\Omega_1(\mathcal{C}^*) &= \Omega_1(\mathcal{C}\mathcal{C}^*\mathcal{Z} + \mathcal{C}^*\mathcal{Z} + \mathcal{C}\mathcal{Z}^* + \mathcal{Z}^*) \\ &= \Omega_1(\mathcal{C})\Omega_0(\mathcal{C}^*\mathcal{Z}) + \Omega_0(\mathcal{C})\Omega_1(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \Omega_1(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \\ &+ \Omega_1(\mathcal{C})\Omega_0(\mathcal{Z}^*) + \Omega_0(\mathcal{C})\Omega_1(\mathcal{Z}^*) + \Omega_1(\mathcal{Z}^*) \\ &= \Omega_1(\mathcal{C})\mathcal{C}\mathcal{Z} + \mathcal{C}\Omega_1(\mathcal{C}^*)\mathcal{Z} + \Omega_1(\mathcal{C}^*)\mathcal{Z} + \\ &+ \Omega_1(\mathcal{C})\mathcal{Z} + \mathcal{C}\mathcal{Z}^*\mathcal{Z} + \mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.21}$$

We used the facts from Lemma 3.1.1. For instance, $\Omega_0(\mathcal{C}^*) = \Omega_0(\mathcal{C}) = \mathcal{C}$. We also used the definitions of $\Omega_1$ and $\Omega_0$, for instance $\Omega_1(\mathcal{Z}^*) = \mathcal{Z}^*\mathcal{Z}$ and $\Omega_0(\mathcal{Z}^*) = \mathcal{Z}$. One term remains undefined on the RHS of (3.21), $\Omega_1(\mathcal{C})$. Since it is the only new class that needs defining, we will skip pushing it to $\mathcal{B}$ and immediatelly popping it out of there. The definition is below.

$$\begin{aligned}\Omega_1(\mathcal{C}) &= \Omega_1(\mathcal{C}\mathcal{C}\mathcal{Z} + 2\mathcal{C}\mathcal{Z} + \mathcal{Z}) \\ &= \Omega_1(\mathcal{C})\mathcal{C}\mathcal{Z} + \mathcal{C}\Omega_1(\mathcal{C})\mathcal{Z} + \mathcal{C}\mathcal{C}\mathcal{Z}^*\mathcal{Z} + 2\Omega_1(\mathcal{C})\mathcal{Z} + 2\mathcal{C}\mathcal{Z}^*\mathcal{Z} + \mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.22}$$

There is no undefined term on the RHS of (3.22) and therefore $\mathcal{B}$ is currently a singleton.

$$\mathcal{B} = \{\Omega_{11}(\overline{\mathcal{C}}^*)\}$$

Therefore, we define the only class in $\mathcal{B}$ first.

$$\begin{aligned}\Omega_{11}(\overline{\mathcal{C}}^*) &= \Omega_{11}(\mathcal{C}^*\mathcal{Z}) \\ &= \Omega_{11}(\mathcal{C}^*)\mathcal{Z} + \Omega_{10}(\mathcal{C}^*)\Omega_{01}(\mathcal{Z}) \\ &= \Omega_{11}(\mathcal{C}^*)\mathcal{Z} + \Omega_{10}(\mathcal{C}^*)(\mathcal{M} + \mathcal{E})\mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.23}$$

We update $\mathcal{B}$ according to the last line above (recall that $\mathcal{M}$ is already defined in (3.19)). The red items are new.

$$\mathcal{B} = \{\Omega_{11}(\mathcal{C}^*), \Omega_{10}(\mathcal{C}^*)\}.$$

Before we proceed, we remark that if an expression is not an argument to any $\Omega_i$, i.e. it is a *top-level expression*, then it can be evaluated. In what follows,

we will immediately evaluate all top-level expressions as far as is convenient. Notice that, because we only apply one layer of $\Omega$ operators to any original class $(\mathcal{C}, \mathcal{C}^*, \mathcal{M}, \mathcal{Z}, \mathcal{Z}^*)$, all our expressions are top-level.

We pop $\Omega_{11}(\mathcal{C}^*)$ out of $\mathcal{B}$ to define it below. The color coding in (3.24) and most of the future calculations is there to help the reader trace the origin of the expressions.

$$\begin{aligned}
\Omega_{11}(\mathcal{C}^*) &= \Omega_{11}(\mathcal{C}\mathcal{C}^*\mathcal{Z} + \mathcal{C}^*\mathcal{Z} + \mathcal{C}\mathcal{Z}^* + \mathcal{Z}^*) \\
&= \Omega_{11}(\mathcal{C}\mathcal{C}^*\mathcal{Z}) + \Omega_{11}(\mathcal{C}^*\mathcal{Z}) + \Omega_{11}(\mathcal{C}\mathcal{Z}^*) + \Omega_{11}(\mathcal{Z}^*) \\
&= \Omega_{11}(\mathcal{C})\Omega_0(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \Omega_0(\mathcal{C})\Omega_{11}(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \\
&+ \Omega_{10}(\mathcal{C})\Omega_{01}(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \Omega_{10}(\mathcal{C})\Omega_\infty(\mathcal{C}^*)\Omega_{01}(\mathcal{Z}) + \\
&+ \Omega_0(\mathcal{C})\Omega_{10}(\mathcal{C}^*)\Omega_{01}(\mathcal{Z}) + \Omega_{11}(\mathcal{C}^*)\Omega_0(\mathcal{Z}) + \Omega_{10}(\mathcal{C}^*)\Omega_{01}(\mathcal{Z}) + \\
&+ \Omega_{11}(C)\Omega_0(\mathcal{Z}^*) + \Omega_0(\mathcal{C})\Omega_{11}(\mathcal{Z}^*) + \Omega_{10}(\mathcal{C})\Omega_{01}(\mathcal{Z}^*) + \\
&+ \Omega_{11}(\mathcal{Z}^*) \\
&= \Omega_{11}(\mathcal{C})\mathcal{C}\mathcal{Z} + \mathcal{C}\Omega_{11}(\mathcal{C}^*)\mathcal{Z} + \Omega_{10}(\mathcal{C})\Omega_{01}(\mathcal{C}^*)\mathcal{Z} + \\
&+ \Omega_{10}(\mathcal{C})\Omega_\infty(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \mathcal{C}\Omega_{10}(\mathcal{C}^*)(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \\
&+ \Omega_{11}(\mathcal{C}^*)\mathcal{Z} + \Omega_{10}(\mathcal{C}^*)(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \\
&+ \Omega_{11}(\mathcal{C})\mathcal{Z} + \mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \\
&+ \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}
\end{aligned} \tag{3.24}$$

Updating $\mathcal{B}$ yields the following list. The red items those that have not yet been defined and occur on the RHS of (3.24).

$$\mathcal{B} = \{\Omega_{10}(\mathcal{C}^*), \Omega_\infty(\mathcal{C}), \Omega_{11}(\mathcal{C}), \Omega_{10}(\mathcal{C}), \Omega_{01}(\mathcal{C}), \Omega_{01}(\mathcal{C}^*)\}$$

Before we proceed, we define $\Omega_\infty(\mathcal{C})$ as, in the scheme of things, it is a trivial task.

$$\begin{aligned}
\Omega_\infty(\mathcal{C}) &= \Omega_\infty(\mathcal{C})\Omega_\infty(\mathcal{C})\Omega_\infty(\mathcal{Z}) + 2\Omega_\infty(\mathcal{C})\Omega_\infty(\mathcal{Z}) + \Omega_\infty(\mathcal{Z})) \\
&= \Omega_\infty(\mathcal{C})^2(\mathcal{M}+\mathcal{E})\mathcal{Z} + 2\Omega_\infty(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + (\mathcal{M}+\mathcal{E})\mathcal{Z}
\end{aligned} \tag{3.25}$$

Notice that $\Omega_\infty(\mathcal{C})$ is essentially class $\mathcal{C}$ where every atom/point is replaced with a nonempty sequence of points. Indeed, the generating function of $\Omega_\infty(\mathcal{C})$ is $C(z/(1-$

$z$), where $C(z)$ is the generating function of $\mathcal{C}$. Having defined $\Omega_\infty(\mathcal{C})$, we can reduce what are essentially duplicities in $\mathcal{B}$ by applying Lemma 3.1.1. This allows us to disregard $\Omega_{01}(\mathcal{C}^*)$ and only keep $\Omega_{01}(\mathcal{C})$. The new $\mathcal{B}$ is below.

$$\mathcal{B} = \{\Omega_{10}(\mathcal{C}^*), \Omega_{11}(\mathcal{C}), \Omega_{10}(\mathcal{C}), \Omega_{01}(\mathcal{C})\}$$

Next, pop $\Omega_{10}(\mathcal{C}^*)$ out of $\mathcal{B}$. To speed up the procedure, we apply Lemma 3.1.1 to the experssions below as soon as we can. This way there will not be any need to post-adjust $\mathcal{B}$.

$$\begin{aligned}\Omega_{10}(\mathcal{C}^*) &= \Omega_{10}(\mathcal{C}\mathcal{C}^*\mathcal{Z} + \mathcal{C}^*\mathcal{Z} + \mathcal{C}\mathcal{Z}^* + \mathcal{Z}^*)\\ &= \Omega_{10}(\mathcal{C})\Omega_\infty(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + \mathcal{C}\Omega_{10}(\mathcal{C}^*)(\mathcal{M}+\mathcal{E})\mathcal{Z}\\ &+ \Omega_{10}(\mathcal{C}^*)(\mathcal{M}+\mathcal{E})\mathcal{Z} +\\ &+ \Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + \mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z} +\\ &+ \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}\end{aligned} \tag{3.26}$$

We do not augment $\mathcal{B}$ at all after this definition as everything on the RHS has either been defined or is already in $\mathcal{B}$. Therefore, we have

$$\mathcal{B} = \{\Omega_{11}(\mathcal{C}), \Omega_{10}(\mathcal{C}), \Omega_{01}(\mathcal{C})\}.$$

We pop $\Omega_{11}(\mathcal{C})$ and define it below.

$$\begin{aligned}\Omega_{11}(\mathcal{C}) &= \Omega_{11}(\mathcal{C}\mathcal{C}\mathcal{Z} + 2\mathcal{C}\mathcal{Z} + \mathcal{Z})\\ &= \Omega_{11}(\mathcal{C})\mathcal{C}\mathcal{Z} + \mathcal{C}\Omega_{11}(\mathcal{C})\mathcal{Z} + \mathcal{C}\mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \Omega_{10}(\mathcal{C})\Omega_{01}(\mathcal{C})\mathcal{Z} +\\ &+ \Omega_{10}(\mathcal{C})\Omega_\infty(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \mathcal{C}\Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} +\\ &+ 2\Omega_{11}(\mathcal{C})\mathcal{Z} + 2\mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + 2\Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} +\\ &+ \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.27}$$

Again, there is nothing in the definition (3.27) which would not be known or already in $\mathcal{B}$. We do not augment $\mathcal{B}$ and it remains as it was.

$$\mathcal{B} = \{\Omega_{10}(\mathcal{C}), \Omega_{01}(\mathcal{C})\}$$

We pop the next item, $\Omega_{10}(\mathcal{C})$.

$$\begin{aligned}\Omega_{10}(\mathcal{C}) &= \Omega_{10}(\mathcal{C}\mathcal{C}\mathcal{Z} + 2\mathcal{C}\mathcal{Z} + \mathcal{Z}) \\ &= \Omega_{10}(\mathcal{C})\Omega_{\infty}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + \mathcal{C}\Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + \\ &+ \mathcal{C}\mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z} + \\ &+ 2\Omega_{10}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z} + 2\mathcal{C}\mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z} + \mathcal{Z}(\mathcal{M}+\mathcal{E})\mathcal{Z}\end{aligned} \tag{3.28}$$

In (3.28) we do not require any new items. In fact, it turns out that $\Omega_{10}(\mathcal{C})$ is computable from its own definition only, an standalone recursion. There is only one class left in $\mathcal{B}$.

$$\mathcal{B} = \{\Omega_{01}(\mathcal{C})\}$$

We pop the last item from $\mathcal{B}$, $\Omega_{01}(\mathcal{C})$.

$$\begin{aligned}\Omega_{01}(\mathcal{C}) &= \Omega_{01}(\mathcal{C}\mathcal{C}\mathcal{Z} + 2\mathcal{C}\mathcal{Z} + \mathcal{Z}) \\ &= \Omega_{01}(\mathcal{C})\mathcal{C}\mathcal{Z} + \Omega_{\infty}(\mathcal{C})\Omega_{01}(\mathcal{C})\mathcal{Z} + \Omega_{\infty}(\mathcal{C})\Omega_{\infty}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + \\ &+ 2\Omega_{01}(\mathcal{C})\mathcal{Z} + 2\Omega_{\infty}(\mathcal{C})(\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z} + (\mathcal{M}+\mathcal{E})\mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.29}$$

As there is nothing in $\mathcal{B}$ and we do not augment it after (3.29), the definition of $\Omega_{11}(\overline{\mathcal{C}}^*)$ is self-contained assuming that we include definitions (3.18), (3.19), (3.20), (3.21), (3.22), (3.23), (3.24), (3.25), (3.26), (3.27), (3.28), and (3.29).

The corresponding calculations are done in Mathematica [Incb] and can be found in `exampleAv321Av21.nb` inside the `scripts` folder at

https://github.com/jsliacan/thesis.

The gerating function of Av(321|21) coincides with the one obtained in Section 2.3.0.2 and we restate it below for convenience.

$$F(z) = -\frac{1-\sqrt{1-4z}+z(-4+\sqrt{1-4z}+\sqrt{1-5z}/\sqrt{1-z})}{2z^2}$$

The first twelve term sof the counting sequence enumerating Av(321|21) are

$$1, 1, 2, 6, 23, 98, 434, 1949, 8803, 39888, 181201, 825201, 3767757.$$

The sequence is in the OEIS [Inca] as A278301.

### 3.3.2 Example: Av(21|21|21)

As before, Av(21) denotes the class of increasing permutations and Av(21|21|21) then refers to a repeated juxtaposition of Av(21) with Av(21), and then the resulting class juxtaposed with Av(21) (on the right). We know that Av(21), and its non-empty version that we call $\mathcal{M}$, are context-free classes and clearly admitting combinatorial specifications that track the right-most point. This example is a very simple yet not completely degenerate case of iterated juxtapositions with at least three cells. We chose to give full details of enumeration to illustrate how our methods work on repeated juxtapositions. While it is entirely possible to enumerate this class by following the algorithmic definitions in Section 3.2, we exploit the slight degeneracies in this example to shorten the write-up.

First of all, we rewrite Av(21|21|21) in terms of $\Omega$ operators (defined in Section 3.1). Because we choose the gridding with gridlines as far left as possible (gridding from the right), every griddable permutation is griddable in one of the following ways: either all three cells are non-empty, or only the leftmost cell is empty, or only the rightmost cell is non-empty, or all three cells are empty. If all three cells are empty, this case is represented by the empty class $\mathcal{E}$. If the leftmost two cells are empty, this is essentially the monotone increasing class $\mathcal{M}$ (non-empty, as before). For the remaining cases, observe that the rightmost juxtaposition does not need to track the rightmost point as nothing will be juxtaposed on its right. Therefore, at the outermost layer, we are only interested in expressions of the form $\Omega_{10}(\mathcal{M})$ and $\Omega_{10}(\mathcal{M}|\mathcal{M})$ (when one and zero cells are empty, respectively). Consider the case when the leftmost cell is empty and the remaining two cells are non-empty. We express this situation by $\Omega_{10}(\mathcal{M}^*)(\mathcal{M}+\mathcal{E})$ — the left of the two cells is a non-empty monotone increasing class whose combinatorial specification traks the rightmost point, $\mathcal{M}^*$. The term $(\mathcal{M}+\mathcal{E})$ makes sure that we allow the right cell to place points above everything in the left cell. We have to correct for this, as we do not use the phantom point and we do not track the rightmost (topmost) point in the rightmost cell. The other case is when all three cells are nonempty. Then we need to use the phantom point and track the

rightmost point in the middle cell. So, the middle cell can have a single point or a sequence of length at least two. Therefore, the first two cells are either $\Omega_1(\overline{\mathcal{M}}^*)$ or $\Omega_{11}(\overline{\mathcal{M}}^*)$. We then apply $\Omega_{10}$ to them. Notice that we do not multiply by $\mathcal{M}+\mathcal{E}$ as we are already using the phantom point. Consequently, we remove the trace of the phantom point by dividing the 3-cell expression by $\mathcal{Z}$. Therefore, the final object that we aim to enumerate is $\mathcal{F}$ below.

$$\mathcal{F} = \mathcal{E} + \mathcal{M} + (\mathcal{M}+\mathcal{E})\Omega_{10}(\mathcal{M}^*) + \frac{1}{\mathcal{Z}}(\Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)) + \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))) \tag{3.30}$$

Notice that $\mathcal{F}$ is not a class or a combinatorial specification of a class. It is merely an expression that is formally correct and captures both structure and enumeration of Av(21|21|21). It is a sort of hybrid benefiting from both the combinatorial specification and the generating function of Av(21|21|21). Therefore, we will aim to obtain the combinatorial specification for the following class

$$\mathcal{E} + \mathcal{M} + (\mathcal{M}+\mathcal{E})\Omega_{10}(\mathcal{M}^*) + \Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)) + \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*)). \tag{3.31}$$

Once we have it, we obtain its generating function and at this stage, we remove the trace of the phantom point by dividing the appropriate terms by $\mathcal{Z}$.

We begin by defining the most simple classes so that we do not have to worry about them later.

$$\mathcal{M} = \mathcal{Z} + \mathcal{M}\mathcal{Z} \tag{3.32}$$
$$\mathcal{M}^* = \mathcal{Z}^* + \mathcal{M}\mathcal{Z}^* \tag{3.33}$$
$$\overline{\mathcal{M}}^* = \mathcal{M}^*\mathcal{Z} \tag{3.34}$$
$$\Omega_0(\mathcal{M}) = \mathcal{Z} + \mathcal{M}\mathcal{Z} = \mathcal{M} \tag{3.35}$$
$$\Omega_0(\mathcal{M}^*) = \mathcal{Z} + \mathcal{M}\mathcal{Z} = \mathcal{M} \tag{3.36}$$

We will often use the fact that $(\mathcal{M}+\mathcal{E})\mathcal{Z} = \mathcal{M}$, such as in (3.35) and (3.36). We

also use it to collapse $\Omega_\infty(\mathcal{Z}^*)$ into $\mathcal{M}$ as follows: $\Omega_\infty(\mathcal{Z}^*) = (\mathcal{M} + \mathcal{E})\mathcal{Z} = \mathcal{M}$.

$$\begin{aligned} \Omega_\infty(\mathcal{M}) &= \Omega_\infty(\mathcal{Z}) + \Omega_\infty(\mathcal{M})\Omega_\infty(\mathcal{Z}) \\ &= \mathcal{M} + \Omega_\infty(\mathcal{M})\mathcal{M} \end{aligned} \tag{3.37}$$

Next, we define the terms with two nonempty cells.

$$\begin{aligned} \Omega_{10}(\mathcal{M}) &= \Omega_{10}(\mathcal{Z}) + \Omega_{10}(\mathcal{M}\mathcal{Z}) \\ &= \Omega_{10}(\mathcal{Z}) + \Omega_{10}(M)\Omega_\infty(\mathcal{Z}) + \Omega_0(\mathcal{M})\Omega_{10}(\mathcal{Z}) \\ &= \mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{M})\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{Z} \end{aligned} \tag{3.38}$$

**Observation 3.3.1.** Notice that $\Omega_{10}$ has the same effect on $\mathcal{M}$ as on $\mathcal{M}^*$, i.e. $\Omega_{10}(\mathcal{M}^*) = \Omega_{10}(\mathcal{M})$. This is quite peculiar sa $\Omega_{10}$ does not ignore the rightmost point in its argument. However, in $\mathcal{M}$ and $\mathcal{M}^*$, the rightmost point is also the topmost point. And hence every point in $\mathcal{M}$ and $\mathcal{M}^*$ is below the rightmost point. By the same reasoning, $\Omega_1(\mathcal{M}^*) = \Omega_1(\mathcal{M})$ and $\Omega_{11}(\mathcal{M}^*) = \Omega_{11}(\mathcal{M})$. We will use these identities below to avoid defining expressions duplicitly.

We have now defined every class needed for the first three terms of (3.31). Next, we define the terms that represent the three nonempty cells. As in Section 3.3.1, let $\mathcal{B}$ be the stack of undefined items. We begin with

$$\mathcal{B} = \{\Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)) + \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))\}.$$

Since we have defined $\overline{\mathcal{M}}^*$ in (3.34), we now need to push $\Omega_1(\overline{\mathcal{M}}^*)$ and $\Omega_{11}(\overline{\mathcal{M}}^*)$ onto the stack before we define anything else. Hence,

$$\mathcal{B} = \{\Omega_1(\overline{\mathcal{M}}^*), \Omega_{11}(\overline{\mathcal{M}}^*), \Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)), \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))\}.$$

We pop the first item and define it below.

$$\begin{aligned} \Omega_1(\overline{\mathcal{M}}^*) &= \Omega_1(\mathcal{M}^*)\Omega_0(\mathcal{Z}) \\ &= \Omega_1(\mathcal{M})\mathcal{Z} \end{aligned} \tag{3.39}$$

Recall that $\Omega_1(\mathcal{M}^*)$ is the same object as $\Omega_1(\mathcal{M})$ thanks to the Observation 3.3.1, that the rightmost point is also the topmost point in $\mathcal{M}$. Therefore, we need to define $\Omega_1(\mathcal{M})$. We skip pushing and popping it onto and from the stack and define it straight away.

$$\begin{aligned}\Omega_1(\mathcal{M}) &= \Omega_1(\mathcal{Z} + \mathcal{M}\mathcal{Z}) \\ &= \Omega_1(\mathcal{Z}) + \Omega_1(\mathcal{M})\Omega_0(\mathcal{Z}) + \Omega_0(\mathcal{M})\Omega_1(\mathcal{Z}) \\ &= \mathcal{Z}^*\mathcal{Z} + \Omega_1(\mathcal{M})\mathcal{Z} + \mathcal{M}\mathcal{Z}^*\mathcal{Z}\end{aligned} \tag{3.40}$$

Currently, $\mathcal{B}$ looks as follows

$$\mathcal{B} = \{\Omega_{11}(\overline{\mathcal{M}}^*), \Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)), \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))\}.$$

We pop the next element and break it down.

$$\begin{aligned}\Omega_{11}(\overline{\mathcal{M}}^*) &= \Omega_{11}(\mathcal{M}^*\mathcal{Z}) \\ &= \Omega_{11}(\mathcal{M}^*)\Omega_0(\mathcal{Z}) + \Omega_{10}(\mathcal{M}^*)\Omega_{01}(\mathcal{Z}) \\ &= \Omega_{11}(\mathcal{M}^*)\mathcal{Z} + \Omega_{10}(\mathcal{M}^*)\mathcal{M}\mathcal{Z}\end{aligned} \tag{3.41}$$

Where the last line follows from the definition of $\Omega_0$ and $\Omega_{01}$ and from the fact that $(\mathcal{M} + \mathcal{E})\mathcal{Z} = \mathcal{M}$. Given that $\Omega_{11}(\mathcal{M}^*) = \Omega_{11}(\mathcal{M})$ and $\Omega_{10}(\mathcal{M}^*) = \Omega_{10}(\mathcal{M})$, we push the only new element $\Omega_{11}(\mathcal{M})$ onto $\mathcal{B}$ (recall that $\Omega_{10}(\mathcal{M})$ was defined in (3.38)).

$$\mathcal{B} = \{\Omega_{11}(\mathcal{M}), \Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)), \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))\}$$

We pop $\Omega_{11}(\mathcal{M})$ next.

$$\begin{aligned}\Omega_{11}(\mathcal{M}) &= \Omega_{11}(\mathcal{Z}) + \Omega_{11}(\mathcal{M})\Omega_0(\mathcal{Z}) + \Omega_0(\mathcal{M})\Omega_{11}(\mathcal{Z}) + \\ &+ \Omega_{10}(\mathcal{M})\Omega_{01}(\mathcal{Z}) \\ &= \mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{11}(\mathcal{M})\mathcal{Z} + \mathcal{M}\mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{10}(\mathcal{M})\mathcal{M}^*\mathcal{Z}\end{aligned} \tag{3.42}$$

We used the definition of $\Omega_{11}$, $\Omega_0$ and $\Omega_{01}$ to evaluate them on the base case input $\mathcal{Z}$. There is no new class on the last line of (3.42) and hence we need not update

$\mathcal{B}$ which now is

$$\mathcal{B} = \{\Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)), \Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))\}.$$

Notice that we now know both inner expressions of the items in $\mathcal{B}$: $\Omega_1(\overline{\mathcal{M}}^*)$ and $\Omega_{11}(\overline{\mathcal{M}}^*)$. It therefore remains to define the action of $\Omega_{10}$ on them. We pop the next item from $\mathcal{B}$.

$$\begin{aligned}\Omega_{10}(\Omega_1(\overline{\mathcal{M}}^*)) &= \Omega_{10}(\Omega_1(\mathcal{M}^*)\Omega_0(\mathcal{Z})) \quad \text{by (3.34)} \\ &= \Omega_{10}(\Omega_1(\mathcal{M}^*))\Omega_\infty(\Omega_0(\mathcal{Z})) \quad \text{by def of } \Omega_{10} \\ &= \Omega_{10}(\Omega_1(\mathcal{M}))\mathcal{M}\end{aligned} \tag{3.43}$$

where the last line follows from $\Omega_\infty(\Omega_0(Z)) = \Omega_\infty(\mathcal{Z}) = \mathcal{M}$ and $\Omega_{10}(\Omega_1(\mathcal{M}^*)) = \Omega_{10}(\Omega_1(\mathcal{M}))$ by Observation 3.3.1. Given that again we only have one new term to be defined, we omit pushing and popping it to and from $\mathcal{B}$. Therefore,

$$\begin{aligned}\Omega_{10}(\Omega_1(\mathcal{M})) &= \Omega_{10}(\Omega_1(\mathcal{Z})) + \Omega_{10}(\Omega_1(\mathcal{M})\Omega_0(\mathcal{Z})) + \Omega_{10}(\Omega_0(\mathcal{M})\Omega_1(\mathcal{Z})) \\ &= \Omega_{10}(\mathcal{Z}^*)\Omega_\infty(\mathcal{Z}) + \Omega_{10}(\Omega_1(\mathcal{M}))\Omega_\infty(\Omega_0(\mathcal{Z}))+ \\ &+ \Omega_{10}(\Omega_0(\mathcal{M}))\Omega_\infty(\mathcal{Z}^*\mathcal{Z}) + \Omega_0(\Omega_0(\mathcal{M}))\Omega_{10}(\mathcal{Z}^*)\Omega_\infty(\mathcal{Z}) \\ &= \mathcal{M}\mathcal{Z}\mathcal{M} + \Omega_{10}(\Omega_1(\mathcal{M}))\mathcal{M} + \Omega_{10}(\mathcal{M})\mathcal{M}\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{Z}\mathcal{M}\end{aligned} \tag{3.44}$$

There are no undefined objects in the last line and hence $\mathcal{B}$ is now a singleton: it remains to define $\Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*))$.

$$\begin{aligned}\Omega_{10}(\Omega_{11}(\overline{\mathcal{M}}^*)) &= \Omega_{10}(\Omega_{11}(\mathcal{M}^*\mathcal{Z})) \\ &= \Omega_{10}(\Omega_{11}(\mathcal{M}^*)\Omega_0(\mathcal{Z})) + \Omega_{10}(\Omega_{10}(\mathcal{M}^*)\Omega_{01}(\mathcal{Z})) \\ &= \Omega_{10}(\Omega_{11}(\mathcal{M}^*))\Omega_\infty(\Omega_0(\mathcal{Z}))+ \\ &+ \Omega_{10}(\Omega_{10}(\mathcal{M}^*))\Omega_\infty(\Omega_{01}(\mathcal{Z})) + \Omega_0(\Omega_{10}(\mathcal{M}^*))\Omega_{10}(\Omega_{01}(\mathcal{Z})) \\ &= \Omega_{10}(\Omega_{11}(\mathcal{M}))\mathcal{M} + \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M})\mathcal{M}+ \\ &+ \Omega_{10}(\mathcal{M})\Omega_{10}(\mathcal{M})\mathcal{M}\end{aligned} \tag{3.45}$$

The first three equalities above follow from the definitions of $\overline{\mathcal{M}}^*$, $\Omega_{11}$, and $\Omega_{10}$, respectively. The last equality evaluates as many expressions as possible. We used facts from Observation 3.3.1: $\Omega_{11}(\mathcal{M}^*) = \Omega_{11}(\mathcal{M})$ and $\Omega_{10}(\mathcal{M}^*) = \Omega_{10}(\mathcal{M})$, as

well as the definitions of $\Omega_0, \Omega_\infty, \Omega_{01}, \Omega_{10}$ on the base case input $\mathcal{Z}$. However, for (3.45) to be well-defined, we need to include the following set of expressions in the combinatorial specification:

$$\mathcal{B} = \{\Omega_{10}(\Omega_{10}(\mathcal{M})), \Omega_{10}(\Omega_{11}(\mathcal{M}))\}.$$

We pop the top item and define it below.

$$\begin{aligned}
\Omega_{10}(\Omega_{10}(\mathcal{M})) &= \Omega_{10}(\mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{M})\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{Z}) \\
&= \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{Z}) + \Omega_0(\mathcal{M})\Omega_{10}(\mathcal{Z}) + \\
&+ \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty) + \Omega_0(\Omega_{10}(\mathcal{M}))\Omega_{10}(\mathcal{M}) + \\
&+ \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{M}\mathcal{Z}) + \Omega_0(\mathcal{M})\Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{Z}) + \\
&+ \Omega_0(\mathcal{M}\mathcal{M})\Omega_{10}(\mathcal{Z}) \\
&= \Omega_{10}(\mathcal{M})\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{Z} + \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M}) + \\
&+ \Omega_{10}(\mathcal{M})\Omega_{10}(\mathcal{M}) + \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{M})\mathcal{M} + \\
&+ \mathcal{M}\Omega_{10}(\mathcal{M})\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{M}\mathcal{Z}
\end{aligned} \tag{3.46}$$

Since everything in (3.46) is either already known or in $\mathcal{B}$, which currently contains only one element

$$\mathcal{B} = \{\Omega_{10}(\Omega_{11}(\mathcal{M}))\}.$$

Let us now define $\Omega_{10}(\Omega_{11}(\mathcal{M}))$.

$$\begin{aligned}\Omega_{10}(\Omega_{11}(\mathcal{M})) &= \Omega_{10}(\mathcal{M}\mathcal{Z}^*\mathcal{Z}) + \Omega_{10}(\Omega_{11}(\mathcal{M})\mathcal{Z}) + \Omega_{10}(\mathcal{M}\mathcal{M}\mathcal{Z}^*\mathcal{Z}) + \\ &+ \Omega_{10}(\Omega_{10}(\mathcal{M})\mathcal{M}^*\mathcal{Z}) \\ &= \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{Z}^*\mathcal{Z}) + \mathcal{M}\Omega_{10}(\mathcal{Z}^*)\Omega_\infty(\mathcal{Z}) + \\ &+ \Omega_{10}(\Omega_{11}(\mathcal{M}))\Omega_\infty(\mathcal{Z}) + \\ &+ \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{M}\mathcal{Z}^*\mathcal{Z}) + \mathcal{M}\Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{Z}^*\mathcal{Z}) + \\ &+ \mathcal{M}\mathcal{M}\Omega_{10}(\mathcal{Z}^*)\Omega_\infty(\mathcal{Z}) \\ &+ \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M}^*\mathcal{Z}) + \Omega_0(\Omega_{10}(\mathcal{M}))\Omega_{10}(\mathcal{M}^*)\Omega_\infty(\mathcal{Z}) \\ &= \Omega_{10}(\mathcal{M})\mathcal{M}\mathcal{M} + \mathcal{M}\mathcal{M}\mathcal{Z}\mathcal{M} + \Omega_{10}(\Omega_{11}(\mathcal{M}))\mathcal{M} + \\ &+ \Omega_{10}(\mathcal{M})\Omega_\infty(\mathcal{M})\mathcal{M}\mathcal{M} + \mathcal{M}\Omega_{10}(\mathcal{M})\mathcal{M}\mathcal{M} + \\ &+ \mathcal{M}\mathcal{M}\mathcal{M}\mathcal{Z}\mathcal{M} + \\ &+ \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M})\mathcal{M} + \Omega_{10}(\mathcal{M})\Omega_{10}(\mathcal{M})\mathcal{M}\end{aligned} \tag{3.47}$$

Now, everything in the last line of (3.47) has already been defined. Hence, with the information in (3.32)–(3.47), we transform (3.31) into the following expression.

$$\begin{aligned}&\mathcal{E} + \mathcal{M} + (\mathcal{M} + \mathcal{E})\Omega_{10}(\mathcal{M}) + \Big(\Omega_{10}(\Omega_1(\mathcal{M}))\mathcal{M} + \\ &+ \Omega_{10}(\Omega_{11}(\mathcal{M}))\mathcal{M} + \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M})\mathcal{M} + \Omega_{10}(\mathcal{M})\Omega_{10}(\mathcal{M})\mathcal{M}\Big)\end{aligned} \tag{3.48}$$

From (3.48) we see that not all our definitions need to be included in the combinatorial specification (3.31). It turns out that some can be skipped as we managed to express their content directly through lower-level operators. In particular, we need to include definitions in our combinatorial specification: $\mathcal{M}$ (3.32), $\Omega_\infty(\mathcal{M})$ (3.37), $\Omega_{10}(\mathcal{M})$ (3.38), $\Omega_{10}(\Omega_1(\mathcal{M}))$ (3.44), $\Omega_{10}(\Omega_{10}(\mathcal{M}))$ (3.46), and $\Omega_{10}(\Omega_{11}(\mathcal{M}))$ (3.47). However, recall that (3.31) and (3.48) do not actually specify the class Av(21|21|21) as they contain phantom points. The following is not a combinatorial specification, strictly speaking, but its formal meaning is valid and captures the class Av(21|21|21). Therefore, the generating function of the following class/object

stores the counting sequence enumerating Av(21|21|21).

$$\mathcal{F} = \mathcal{E} + \mathcal{M} + (\mathcal{M} + \mathcal{E})\Omega_{10}(\mathcal{M}) + \frac{1}{\mathcal{Z}}\Big(\Omega_{10}(\Omega_1(\mathcal{M}))\mathcal{M} + \\ + \Omega_{10}(\Omega_{11}(\mathcal{M}))\mathcal{M} + \Omega_{10}(\Omega_{10}(\mathcal{M}))\Omega_\infty(\mathcal{M})\mathcal{M} + \Omega_{10}(\mathcal{M})\Omega_{10}(\mathcal{M})\mathcal{M}\Big) \tag{3.49}$$

The relevant Mathematica script implementing enumeration of Av(21|21|21) via the process described in this section can be found in `exampleMMM.nb`. The file resides in the `scripts` folder of the accompanying thesis repository at

https://github.com/jsliacan/thesis.

The generating function of Av(21|21|21) is

$$F(z) = \frac{22x^5 - 52x^4 + 56x^3 - 32x^2 + 9x - 1}{(x-1)^3(2x-1)^2(3x-1)}. \tag{3.50}$$

The counting sequence that we obtain for the number of permutations in Av(21|21|21) of length $k = 0, \ldots, 12$ is

$$1, 1, 2, 6, 23, 93, 360, 1312, 4541, 15111, 48854, 154674, 482355 \ldots$$

This agrees with Bevan's enumeration of Av(21|21|21) in his thesis [Bev15b], Part I, Table 3.1. The sequence is not in the OEIS [Inca].

### 3.3.3 Example: Separable next to monotone

The class of separable permutations has finitely many simple permutations and is relatively simple. We still think this example is useful in that it demonstrates that our method can be used to enumerate various juxtapositions exactly. To the best of our knowledge, the juxtaposition class $\mathcal{S}|\mathcal{M}$, where $\mathcal{M}$ is an increasing monotone class, has not been enumerated yet. We juxtapose $\mathcal{M}$ on the right of the class of separable permutations $\mathcal{S}$ and choose to work with the following combinatorial

specification of $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S}^* &= \mathcal{Z}^* + \mathcal{S}_\oplus \mathcal{S}^* + \mathcal{S}^* \mathcal{S}_\ominus \\
\mathcal{S} &= \mathcal{Z} + \mathcal{S}_\oplus \mathcal{S} + \mathcal{S}\mathcal{S}_\ominus \\
\mathcal{S}_\ominus &= \mathcal{Z} + \mathcal{S}_\oplus \mathcal{S} \\
\mathcal{S}_\oplus &= \mathcal{Z} + \mathcal{S}\mathcal{S}_\ominus.
\end{aligned} \tag{3.51}$$

We also know that

$$\begin{aligned}
\mathcal{M} &= \mathcal{Z} + \mathcal{M}\mathcal{Z} \\
\mathcal{M}^* &= \mathcal{Z}^* + \mathcal{M}\mathcal{Z}^*
\end{aligned} \tag{3.52}$$

Before we proceed with the enumeration, notice that whether $\mathcal{M}$ is a monotone increasing or a monotone decreasing class, $\mathcal{S}|\mathcal{M}$ is enumerated by the same generating function. This is obvious from the chosen combinaorial specification (3.51). Indeed, if $\mathcal{M}$ is monotone decreasing, we flip the entire juxtaposition around a horizontal axis (so that it is upside down). The main observation is that if $P \in \mathcal{S}$, then a "flip" of $P$ around the horizontal axis is also in $\mathcal{S}$. This is easy to verify. Therefore, in case of decreasing $\mathcal{M}$, we enumerate the upside down juxtaposition.

Back to the example with a monotone increasing $\mathcal{M}$. To make this example as short as possible, we will not write out the whole derivation of expressions in the combinatorial specification. It is a routine process which could, in principle, be automated. Also, instead of keeping track of a set $\mathcal{B}$ of classes that we need to define, we will determine the whole list of classes that we need ahead of time. Then we just define the classes in that list.

Notice that we will not need to define $\mathcal{S}^*_\oplus$ or $\mathcal{S}^*_\ominus$. This is because $\mathcal{S}_\oplus$ and $\mathcal{S}_\ominus$, the way they are used in (3.51), can never contain the rightmost point. Refer to the pictorial definition (3.4) of $\mathcal{S}$ for clearer image. Moreover, notice that

$$\begin{aligned}
\Omega_0(\mathcal{S}^*) &= \Omega_0(\mathcal{S}) \\
\Omega_\infty(\mathcal{S}^*) &= \Omega_\infty(\mathcal{S}) \\
\Omega_{01}(\mathcal{S}^*) &= \Omega_{01}(\mathcal{S}).
\end{aligned}$$

All of these operators ignore and erase the rightmost points of their arguments. Hence, it does not matter if we feed them $\mathcal{S}^*$ or $\mathcal{S}$. Moreover, $\Omega_0(\mathcal{S}) = \mathcal{S}$, and therefore $\Omega_0(\mathcal{S}^*) = \mathcal{S}$ as well. We are left with Table 3.1 of items (combinations of arguments and operators) that we need to define in the combinatorial specification fo $\mathcal{S}$.

| | $\mathcal{S}$ | $\mathcal{S}^*$ | $\mathcal{S}_\ominus$ | $\mathcal{S}_\oplus$ |
|---|---|---|---|---|
| $\Omega_0$ | ✗ | ✗ | ✗ | ✗ |
| $\Omega_\infty$ | | ✗ | | |
| $\Omega_1$ | | | | |
| $\Omega_{10}$ | | | | |
| $\Omega_{11}$ | | | | |
| $\Omega_{01}$ | | ✗ | | |

Table 3.1: The positions with ✗ mark the combinations (operator-argument) which we do not need to define in the combinatorial specification of $\mathcal{S}|\mathcal{M}$ because we either know them already or they amount to the same output as some other combinations.

We are looking to enumerate $\mathcal{F}$, which is just $\mathcal{S}|\mathcal{M}$ rewritten in language of $\Omega$ operators.

$$\mathcal{F} = \mathcal{E} + \mathcal{M} + (\Omega_1(\mathcal{S}^*) + \Omega_{11}(\mathcal{S}^*))(\mathcal{M} + \mathcal{E}) \tag{3.53}$$

Clearly, according to the number of empty cells, we have three cases. The case when both cells are empty is represented by $\mathcal{E}$. If only one cell is empty, then it must be the left cell because our choice of gridding places the gridline as far left as possible. The remaining cell must then be non-empty and monotone increasing, or $\mathcal{M}$. If both cells are non-empty, then there is either a single point on the right-hand side, represented by $\Omega_1(\mathcal{S}^*)$, or there are at least two points on the right-hand side, represented by $\Omega_{11}(\mathcal{S}^*)$. In both these cases we need to allow points on the right-hand side to be above all points on the left-hand side. This is achieved by the term $(\mathcal{M}+\mathcal{E})$. Notice that we did not need to use phantom points as we juxtapose the monotone class only once. This shortcut does not generalise to iterated juxtapositions and we already used it in Section 3.3.2 for $\mathcal{M}|\mathcal{M}$ as the first juxtaposition in $\mathcal{M}|\mathcal{M}|\mathcal{M}$.

Before we proceed, let us recall that all operators are linear. Let us begin by

defining the action of $\Omega_\infty$.

$$\begin{aligned}
\Omega_\infty(\mathcal{S}) &= \mathcal{M} + \Omega_\infty(\mathcal{S}_\oplus)\Omega_\infty(\mathcal{S}) + \Omega_\infty(\mathcal{S})\Omega_\infty(\mathcal{S}_\ominus) \\
\Omega_\infty(\mathcal{S}_\ominus) &= \mathcal{M} + \Omega_\infty(\mathcal{S}_\oplus)\Omega_\infty(\mathcal{S}) \\
\Omega_\infty(\mathcal{S}_\oplus) &= \mathcal{M} + \Omega_\infty(\mathcal{S})\Omega_\infty(\mathcal{S}_\ominus)
\end{aligned} \tag{3.54}$$

This deals with the second row of the Table 3.1. We define $\Omega_1$ next.

$$\begin{aligned}
\Omega_1(\mathcal{S}) &= \mathcal{Z}^*\mathcal{Z} + \Omega_1(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_1(\mathcal{S}) + \Omega_1(\mathcal{S})\mathcal{S}_\ominus + \mathcal{S}\Omega_1(\mathcal{S}_\ominus) \\
\Omega_1(\mathcal{S}^*) &= \mathcal{Z}^*\mathcal{Z} + \Omega_1(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_1(\mathcal{S}^*) + \Omega_1(\mathcal{S}^*)\mathcal{S}_\ominus \\
\Omega_1(\mathcal{S}_\oplus) &= \mathcal{Z}^*\mathcal{Z} + \Omega_1(\mathcal{S})\mathcal{S}_\ominus + \mathcal{S}\Omega_1(\mathcal{S}_\ominus) \\
\Omega_1(\mathcal{S}_\ominus) &= \mathcal{Z}^*\mathcal{Z} + \Omega_1(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_1(\mathcal{S})
\end{aligned} \tag{3.55}$$

This deals with the third row in Table 3.1. The next operator we define is $\Omega_{10}$.

$$\begin{aligned}
\Omega_{10}(\mathcal{S}) &= \mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{S}_\oplus)\Omega_\infty(\mathcal{S}) + \mathcal{S}_\oplus\Omega_{10}(\mathcal{S}) + \Omega_{10}(\mathcal{S})\Omega_\infty(\mathcal{S}_\ominus) + \\
&\quad + \mathcal{S}\Omega_{10}(\mathcal{S}_\ominus) \\
\Omega_{10}(\mathcal{S}^*) &= \mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{S}_\oplus)\Omega_\infty(\mathcal{S}^*) + \mathcal{S}_\oplus\Omega_{10}(\mathcal{S}^*) + \Omega_{10}(\mathcal{S}^*)\Omega_\infty(\mathcal{S}_\ominus) \\
\Omega_{10}(\mathcal{S}_\oplus) &= \mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{S})\Omega_\infty(\mathcal{S}_\ominus) + \mathcal{S}\Omega_{10}(\mathcal{S}_\ominus) \\
\Omega_{10}(\mathcal{S}_\ominus) &= \mathcal{M}\mathcal{Z} + \Omega_{10}(\mathcal{S}_\oplus)\Omega_\infty(\mathcal{S}) + \mathcal{S}_\oplus\Omega_{10}(\mathcal{S})
\end{aligned} \tag{3.56}$$

This deals with the fourth row of Table 3.1. The operator $\Omega_{11}$ is next.

$$\begin{aligned}
\Omega_{11}(\mathcal{S}) &= \mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{11}(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_{11}(\mathcal{S}) + \Omega_{10}(\mathcal{S}_\oplus)\Omega_{01}(\mathcal{S}) + \\
&\quad + \Omega_{11}(\mathcal{S})\mathcal{S}_\ominus + \mathcal{S}\Omega_{11}(\mathcal{S}_\ominus) + \Omega_{10}(\mathcal{S})\Omega_{01}(\mathcal{S}_\ominus) \\
\Omega_{11}(\mathcal{S}^*) &= \mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{11}(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_{11}(\mathcal{S}^*) + \Omega_{10}(\mathcal{S}_\oplus)\Omega_{01}(\mathcal{S}^*) + \\
&\quad + \Omega_{11}(\mathcal{S}^*)\mathcal{S}_\ominus + \Omega_{10}(\mathcal{S}^*)\Omega_{01}(\mathcal{S}_\ominus) \\
\Omega_{11}(\mathcal{S}_\oplus) &= \mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{11}(\mathcal{S})\mathcal{S}_\ominus + \mathcal{S}\Omega_{11}(\mathcal{S}_\ominus) + \Omega_{10}(\mathcal{S})\Omega_{01}(\mathcal{S}_\ominus) \\
\Omega_{11}(\mathcal{S}_\ominus) &= \mathcal{M}\mathcal{Z}^*\mathcal{Z} + \Omega_{11}(\mathcal{S}_\oplus)\mathcal{S} + \mathcal{S}_\oplus\Omega_{11}(\mathcal{S}) + \Omega_{10}(\mathcal{S}_\oplus)\Omega_{01}(\mathcal{S})
\end{aligned} \tag{3.57}$$

This defines the row five of Table 3.1. It now remains to define $\Omega_{01}$.

$$\begin{aligned}\Omega_{01}(\mathcal{S}) &= \mathcal{M}^*\mathcal{Z} + \Omega_{01}(\mathcal{S}_\oplus)\mathcal{S} + \Omega_\infty(\mathcal{S}_\oplus)\Omega_{01}(\mathcal{S}) + \\ &\quad + \Omega_{01}(\mathcal{S})\mathcal{S}_\ominus + \Omega_\infty(\mathcal{S})\Omega_{01}(\mathcal{S}_\ominus) \\ \Omega_{01}(\mathcal{S}_\ominus) &= \mathcal{M}^*\mathcal{Z} + \Omega_{01}(\mathcal{S}_\oplus)\mathcal{S} + \Omega_\infty(\mathcal{S}_\oplus)\Omega_{01}(\mathcal{S}) \\ \Omega_{01}(\mathcal{S}_\oplus) &= \mathcal{M}^*\mathcal{Z} + \Omega_{01}(\mathcal{S})\mathcal{S}_\ominus + \Omega_\infty(\mathcal{S})\Omega_{01}(\mathcal{S}_\ominus)\end{aligned} \tag{3.58}$$

The combinatorial specification describing $\mathcal{S}|\mathcal{M}$ involves all terms from Table 3.1 together with $\mathcal{M}, \mathcal{M}^*, \mathcal{S}$ and $\mathcal{S}^*$. One can check that there is no undefined term on the RHS of any of the items in Table 3.1 — meaning that every term used on the RHS of any one of the equations is defined elsewhere in the combinatorial specification. Let $F(z)$ be the generating function of $\mathcal{F}$ (thus of $\mathcal{S}|\mathcal{M}$). Since $F(z)$ is not sufficiently compact to be given here in full, we resort to only listing the first twelve terms of the counting sequence of $\mathcal{S}|\mathcal{M}$. They are below.

$$1, 1, 2, 6, 24, 115, 609, 3409, 19728, 116692, 701062, 4261581, 26146111.$$

The sequence is not on the OEIS [Inca]. The accompanying Mathematica [Incb] file can be found in the `scripts` folder as `exampleSeparable.nb` at:

`https://github.com/jsliacan/thesis/`.

## 3.4 Conclusion

Let us first summarise what the yields of this chapter are and then we point out several obvious and powerful ways to continue exploring the topic. Section 3.1 outlines what is known about enumerating grid classes. We contribute by describing a way for enumerating any $1 \times n$ monotone grid class one of whose cells has been replaced by an arbitrary context-free class (that admits point tracking as described in Section 3.1). Apart from the framework for producing exact enumeration results for these classes, we also prove that this general setting is as nice as possible — that any such class admits a generating function that is algebraic. Recall that context-free classes are often enumerated by counting sequences whose generating functions are algebraic non-rational. Moreover, we give an example of

a large family of context-free permutation classes which satisfy the conditions we set in Theorem 3.2.1 — the context-free classes with finitely many simple permutations.

We demonstrate how to use this general framework for exact enumeration of specific permutation classes. We do this by enumerating three key permutation classes.

The example in Section 3.3.1 uses our $\Omega$ operators to enumerate Av(321|21) (also enumerated in Chapter 2 via other means). What makes this example interesting is that Av(321) contains infinitely many simples and therefore, in some sense, we require the full strength of Theorem 3.2.1 to deal with this example.

The next example, in Section 3.3.2, illustrates how our framework deals with repeated juxtapositions. This was the whole point of our work from the beginning and this example shows how straightforward the task became with $\Omega$ operators. Additionally, Bevan [Bev15b] enumerates this class in his thesis and our method covers that case too.

The last example, in Section 3.3.3, demonstrates the enumeration of separable permutations juxtaposed with a monotone increasing class. The class of separable permutations contains finitely many simple permutations (zero, to be exact). The enumeration of this class is of interest for multiple reasons. Firstly, separable permutations are in one-to-one correspondence with Schröder paths (lattice paths that stay below the diagonal and allows steps $(1,0)$, $(0,1)$, and $(1,1)$) — a generalization of Dyck paths. In other words, juxtaposing a monotone class next to the class of separable permutations is a natural next step after Catalan juxtapositions in Chapter 2. Secondly, separable permutations are the least complex non-degenerate class with finitely many simples, thereby serving as an example for Corollary 3.2.1. Thirdly, to the best of our knowledge, the juxtaposition of separable permutations with a monotone class has not been enumerated before. Therefore, this is an example that is also a new result at the same time.

Even though the three examples above were chosen carefully to fit well with the existing work, their main purpose is to illustrate the practical side of the machinery leading to Theorem 3.2.1 — the main result of this chapter.

When defining $\Omega$ operators in Section 3.1, we present the definitions as recursions in functional form. Assuming that the user constructs on her own the expression $\mathcal{F}$ — the one which represents the juxtaposition in question as a disjoint union of juxtapositions of non-empty cells, and whose generating function enumerates the desired juxtaposition — the rest of the task towards obtaining the generating function and the counting sequence of the desired juxtaposition class lends itself to automation. In other words, after parsing the input from the user, the rest is an algorithmic task, and hence implementable on a PC. This would be a very useful tool for enumerating small juxtaposition classes of the form that we described. We do plan to address this in the near future.

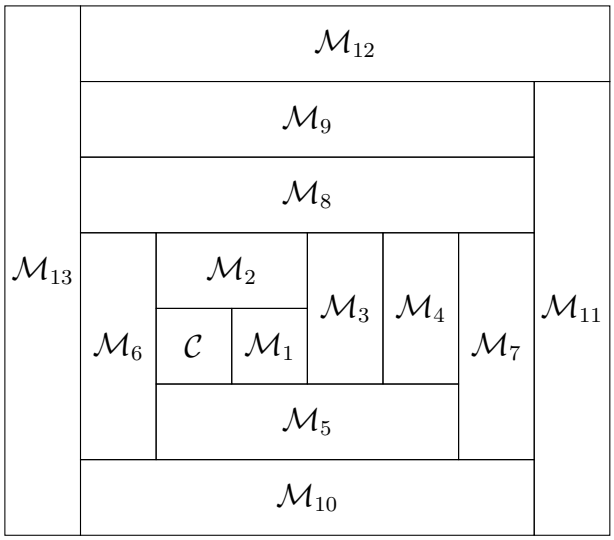



Figure 3.14: An example of unrestricted iterative juxtaposition of thirteen monotone classes $\mathcal{M}_1, \ldots, \mathcal{M}_{13}$ onto a context-free class $\mathcal{C}$.

The one major direction of research that suggests itself after our work is the generalisation to juxtapositions from all four sides: top, bottom, right, and left. See Figure 3.14. This would require tracking all four extremal points in the permutation. This in itself is not an easy task, as it would require two mutually dependent combinatorial specifications. One that tracks the vertical position of points and the other one that tracks their horizontal positions. Both horizontal and vertical positions of each point are needed to deal with top and side juxta-

positions. Assuming we can deal with that, it still remains to determine how to uniquely grid any one given permutation (provided that it is griddable into a given class).

Notice that the grid classes like the one in Figure 3.14 contain all acyclic grid classes with one context-free cell. Therefore, a result along the lines of Theorem 3.2.1 about these acyclic classes is among the goals of our continued work in this area. Another natural question is whether the respective results translate to regular class $\mathcal{C}$ juxtaposed with monotone classes, i.e. are such juxtapositions regular? Do they have rational generating functions?

Finally, we would like to point out that a context-free class $\mathcal{C}$ was chosen because it is a "nicely behaved" object to juxtapose monotone classes next to. However, other objects — maybe more general ones — could be manageable under favourable circumstances. It remains a direction to pursue.

# Part II

# Packing

# Chapter 4

# Packing small permutations

We consolidate what is currently known about packing densities of 4-point permutations and in the process improve the lower bounds for the packing densities of 1324 and 1342. We also provide rigorous upper bounds for the packing densities of 1324, 1342, and 2413. All our bounds are within $10^{-4}$ of the true packing densities. Together with the known bounds, this gives us a fairly complete picture of all 4-point packing densities. We also list a number of upper bounds for small permutations of length at least five. Our main tool for the upper bounds is the framework of flag algebras introduced by Razborov in 2007.

## 4.1 Introduction

In this paper, we study packing densities of small permutations. A *permutation* is an ordered tuple utilizing all integers from $\{1, \ldots, n\}$. We say that $S = S[1]S[2]\cdots S[m]$ is a *sub-permutation* of $P = P[1]P[2]\cdots P[n]$ if there exists an $m$-subset $\{k_1, \ldots, k_m\}$ of $\{1, \ldots, n\}$ such that for all $1 \leq i, j \leq m$, $S[i] < S[j]$ whenever $P[k_i] < P[k_j]$. We denote by $\#(S, P)$ the number of occurrences of $S$ as a sub-permutation of $P$. Let $\mathcal{P}_n$ be the set of all permutations of length $n$. If $\#(S, n) = \max_{P \in \mathcal{P}_n} \#(S, P)$, then the *packing density* of $S$ is defined to be $p(S) = \lim_{n\to\infty} \#(S, n)/\binom{n}{m}$.

The study of permutation packing densities began with Wilf's 1992 SIAM address. Galvin (unpublished) soon rediscovered the averaging argument of [KNS64],

| S | lower bound | ref LB | upper bound | ref UB |
|---|---|---|---|---|
| 1234 | 1 | trivial | 1 | trivial |
| 1432 | $\beta$ | [Pri97] | $\beta$ | [Pri97] |
| 2143 | 3/8 | trivial | 3/8 | [Pri97] |
| 1243 | 3/8 | trivial | 3/8 | [AAH+02] |
| 1324 | 0.244* | [Pri97] | −* | [Pri97] |
| 1342 | $\gamma$* | [Bat] | 0.1988373* | [BHL+15] |
| 2413 | $\approx 0.104724$ | [PS10] | 0.1047805* | [BHL+15] |

Table 4.1: Overview of packing densities for 4-point permutations. Values $\beta$ and $\gamma$ are known exactly: $\beta = 6\sqrt[3]{\sqrt{2}-1} - 6/\sqrt[3]{\sqrt{2}-1} + 4 \sim 0.423570$, $\gamma = (2\sqrt{3}-3)\beta \sim 0.19657960$. We know that the packing density of 1324 is close to 0.244 but there is no non-trivial upper bound. The items with an (*) asterisk will be updated by the current work.

thus proving that $p(S)$ exists for all permutations $S$. The original argument was in the setting of graph theory. In 1993, Stromquist, and independently Galvin and Kleitman (both unpublished), found the packing density of 132. Up to symmetry, 132 is the only permutation of length 3 with a non-trivial packing density.

For 4-point permutations and their packing densities, it is useful to consult Table 4.1. First results for 4-point permutations, including 1324, 1432, and 2143, came as part of the investigation of various *layered patterns* by [Pri97]. Later, [AAH+02] proved a tight upper bound for 1243, and upper bounds of 2/9 for both 2413 and 1342. The current lower bound for the packing density of 2413 was given by [PS10]. The upper bounds of 0.1047805 and 0.1988373 for 2413 and 1342, respectively, are mentioned in passing in [BHL+15]. They do not discuss them any further.

It is worthwhile to point out that [BHL+15] used flag algebras to attack the packing density problem for monotone sequences of length 4. To the best of our knowledge, the only other application of flag algebras to permutation packing, although indirect, is by [FRV13]. They obtained the inducibility (as packing density is refered to in graph theory) of a 2-star directed graph  . Their result implies the known upper bound for the packing density of 132. Later, [Hua14] used an argument exploiting equivalence classes of vertices to extend the result to all directed $k$-stars. This argument was known in the permutations setting since [Pri97] used it to establish the packing densities $p(1k\ldots2)$ for all $k$. Similarly, although flag

algebra software package Flagmatic, written by [Vau13], has been available since 2013, it has not previously been used to obtain an upper bound on the packing density of 1324.

Therefore, we decided to use the flag algebras method to collect, enhance, and improve results in permutation packing densities. In addition to the mathematical content, we make available a flag algebras package for permutations, Permpack, written as a Sage script. For more information about the software, follow [Dev17]. It does all our computations and can be used for further research. Permpack uses syntax similar to Flagmatic, but requires no installation. We hope this makes it more user friendly.

The rest of this paper is structured as follows. The aim of Section 4.2 is to introduce notation and concepts, including the part of flag algebras that we need. While [Raz07] presented flag algebras in the general setting of a universal model theory without constants and function symbols, we choose permutations to be the structures on which we base our exposition. Section 4.3 presents the main results of this chapter. We use flag algebras to provide upper bounds for the packing densities of 4-point permutations 1324, 1342, and 2413. We learnt belatedly about the existence of the latter two bounds from [BHL+15]. Regarding lower bounds, we give a new lower bound construction for the packing density of 1342 that meets our upper bound to within $10^{-5}$. In case of 1324, we provide a lower bound that agrees with the upper bound on the first five decimal places. Section 4.4 gives a list of selected upper bounds to illustrate the potential of the flag algebras method in the area of permutation packing. These results are not best possible, but can be obtained effortlessly by using our flag algebras package Permpack.

## 4.2 Definitions and concepts

Given $S$ and $P$ of lengths $m$ and $n$, respectively, we let $\#(S, P)$ denote the number of times that $S$ occurs as a subpermutation of $P$. The *density* of $S$ in $P$ is

$$p(S, P) = \frac{\#(S, P)}{\binom{n}{m}}.$$

If $n < m$, we set $p(S, P) = 0$. Intuitively, $p(S, P)$ is the probability that a random $m$-set of positions from $[n]$ induces a pattern in $P$ that is order-isomorphic to $S$. For example, $p(12, 132) = 2/3$ as both 13 and 12 are order-isomorphic to 12 while 32 is not.

Recall that if $\mathcal{F}$ is a set of *forbidden* permutations. We say that permutation $P$ is $\mathcal{F}$*-free* if $\#(F, P) = 0$ for all $F \in \mathcal{F}$. Such $P$ is also said to *avoid* $\mathcal{F}$ or be *admissible*. We denote by $\mathcal{P}_n$ the set of all *admissible* permutations of length $n$. It will always be clear from context what $\mathcal{F}$ is. If $\mathcal{F} = \emptyset$, then the admissible set $\mathcal{P}_n$ is the set of all permutations of length $n$. Notice that if $P$ is admissible, then so are all its subpermutations. Most of the work in this paper concerns the case when $\mathcal{F} = \emptyset$. However, the setting remains the same whenever $\mathcal{F}$ is non-empty, and we provide a few examples to this effect.

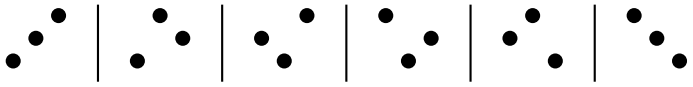


Figure 4.1: Permutations in $\mathcal{P}_3$, with $\mathcal{F} = \emptyset$. From left to right: 123, 132, 213, 312, 231, 321.

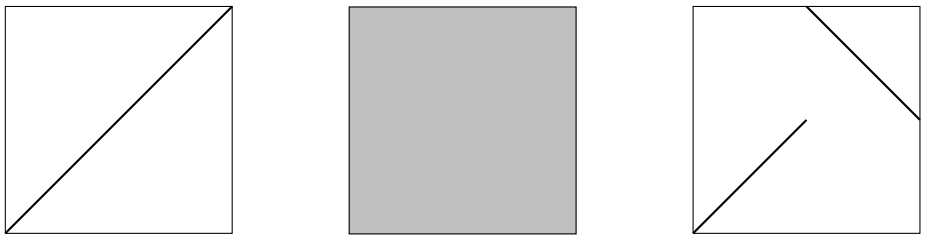
Let $P \in \mathcal{P}_n$ and $S \in \mathcal{P}_m$ be admissible permutations, and assume $m \leq n$. The maximum value of $p(S, P)$ over $P \in \mathcal{P}_n$ is denoted by $p(S, n)$. Conversely, a permutation $P$ such that $p(S, P) = p(S, n)$ is an $S$*-maximiser* of length $n$. It is well-known that for every $S$, the sequence $(p(S, n))_{n \geq 0}$ converges to a value in $[0, 1]$ because it is non-increasing and stays between 0 and 1. See [KNS64]. We are now ready to define the quantity that we study, packing density.

**Definition 4.2.1.** Let $S$ be a fixed permutation and $\mathcal{P} = \cup_{n \geq 1} \mathcal{P}_n$ the set of admissible permutations. The *packing density* of $S$ is

$$p(S) = \lim_{n \to \infty} p(S, n).$$

For example, the packing density of 12 in 123-free permutations is $1/2$. Notice that every maximiser of size $n$ has at most two layers. It is then easy to see that they should be of balanced sizes for the packing density to be maximised, i.e. $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$. Let $P_n$ be such balanced 2-layered maximizer of length $n$. Clearly, $p(12, P_n) \to 1/2$ as $n \to \infty$.

We now formalise the ideas about asymptotic quantities and objects that the discussion is leading to. Let $(P_n)_n = P_1, P_2, P_3, \ldots$ be a sequence of permutations of increasing lengths. We say that $(P_n)_n$ is *convergent* if for every permutation $S$, $(p(S, P_n))_{n=1}^{\infty}$ converges. A *permuton* $\mu$ is a probability measure with uniform marginals on the Borel $\sigma$-algebra $\mathcal{B}([0,1]^2)$, i.e. for every $a, b \in [0,1]$ with $a < b$, it holds that $\mu([a,b] \times [0,1]) = b - a = \mu([0,1] \times [a,b])$. See examples of permutons in Figure 4.2.



(a) Increasing (b) Lebesgue (c) 1243-maximiser

Figure 4.2: Examples of permutons. In (a) we have the limit of $(1 \ldots n)_{n=1}^{\infty}$, in (b) it is, with probability one, the limit of a sequence of randomly chosen permutations of each length, and in (c) we have the limit of $(1 \ldots \lfloor n/2 \rfloor n \ldots \lceil n/2 \rceil)_{n=1}^{\infty}$.

Let $\mu$ be a permuton and $S$ a permutation on $[m]$. One can sample $m$ points from $[0,1]^2$ according to $\mu$ and with probability one they will be in general position (no two aligned vertically or horizontally). We define $p(S, \mu)$ as the probability that a randomly sampled $m$ points from $[0,1]^2$ according to $\mu$ are order-isomorphic to $S$. It turns out that every convergent sequence of permutations has its permuton and vice versa. In particular, [HKM+13] proved that for every $(P_n)_{n \geq 0}$ there exists a unique permuton $\mu$ such that for every $S$, $p(S, \mu) = \lim_{n \to \infty} p(S, P_n)$. In this sense, $\mu$ is the limit of the sequence $(P_n)_n$. In the other direction, they proved that if $\mu$ is a permuton and $P_n$ is a permutation of length $n$ sampled at random according to $\mu$ from $[0,1]^2$, then with probability one the sequence $(P_n)_n$ is convergent (with $\mu$ as its limit). The concept of permutation limits was known as "packing measures" since [PS10] used them for constructing the 2413 lower bound. In the current work, we use permutons mainly to describe extremal constructions that yield our lower bounds.

### 4.2.1 Flag Algebras

The term *flag algebras* refers to a framework first introduced by [Raz07]. It proved to be a very useful tool for researchers in extremal graph theory, but found use in other fields as well. For an overview of results aided by flag algebras, see Razborov's own survey [Raz13]. For more extensive expositions, see the PhD theses of [Spe12] and [Vol14]. By now, there are also many papers with explanations and examples such as [BT11], [FRMPV15], [FRV12], [FRV13]. For a long list of important results across disciplines of discrete mathematics that were aided by flag algebras, see the abovementioned theses, especially Chapter 1 of [Vol14]. The main flag algebra result in permutations is [BHL+15]. In their work on quasirnadom permutations, [KP13] mention flag algebras as another way to think about the subject. It is important to note that the method of flag algebras has evolved from other combinatorial and analytic methods in combinatorics which had been used by researchers for a long time. The Cauchy-Schwarz type arguments can be found in e.g. work by [Bon97] as early as 1990s. The ideas pertinent to quasirandomness have been around since [CGW88]. And while there are other analytic methods that were used successfully to attack extremal problems in combinatorics, the method of flag algebras is syntactical and lends itself to automation. The syntax-based nature of flag algebras is the main feature that distinguishes the theory of flag algebras from the theory of dense graph limits (see e.g. [Lov12]). The crux of the method is a systematic conversion of the combinatorial problem into a semidefinite programming problem. The latter can be solved (efficiently) by current SDP solvers. The numerical values returned by the SDP solvers then need to be transformed to exact values (rational or algebraic) to provide valid upper bounds on packing densities.

Before we delve into the method itself, let us consider an example from before. For the remainder of this section, assume that all objects (permutations, flags, types) are admissible unless stated otherwise. Now, assume that we are looking for 123-free permutations $P$ that are as 12-dense as possible. We get the following bound without much effort.

$$
\begin{aligned}
p(12,P) &= \underbrace{p(12,123)p(123,P)}_{=0} + p(12,132)p(132,P) + p(12,213)p(213,P) \\
&\quad + p(12,231)p(231,P) + p(12,312)p(312,P) + p(12,321)p(321,P) \qquad (4.1) \\
&\leq \max\left\{\frac{2}{3},\frac{2}{3},\frac{1}{3},\frac{1}{3},0\right\} = \frac{2}{3}
\end{aligned}
$$

This is strictly better than a trivial bound of 1. However, observe that there is no $P$ of length greater than 4 such that $p(132,P)+p(213,P)=1$. This follows from Erdős-Szekeres theorem (adapted) which states that a permutation of length $(r-1)(s-1)+1$ contains either an increasing subpermutation of length $r$ or a decreasing subpermutation of length $s$. Hence, a permutation of length 5 contains 123 or 321 as a subpermutation. So there are always subsets of size 3 in $P$ which do not induce 132 or 213. Therefore, the bound of $2/3$ is unachievable in practice. Knowing this, it would be useful to be able to control how copies of small permutations, such as 132 and 213, interact inside larger permutations. The method of flag algebras helps us systematically take into account the ways in which small patterns overlap inside larger structures. This takes the form of extra coefficients in front of $p(12,P)$ terms in (4.1). If chosen well, they shift weight away from the large values like $p(12,231)$ and $p(12,312)$ and thereby reduce the maximum over all of them.

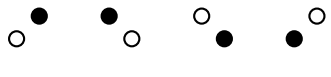
In general, the process is analogous to the example above. If $S$ is a small permutation whose packing density we seek to determine, we pick a reasonably small value $N \geq |S|$. The crude bound then looks as follows.

$$
\begin{aligned}
p(S,P) &= \sum_{P' \in \mathcal{P}_N} p(S,P')p(P',P) \\
&\leq \max_{P' \in \mathcal{P}_N} p(S,P') \qquad (4.2)
\end{aligned}
$$

Before we describe how exactly we leverage overlaps between small patterns, we need to define flags, types, and operations on them.

**Definition 4.2.2** (Flag)**.** A permutation $\tau$-*flag* $S^\tau$ is a permutation $S$ together with a distinguished subpermutation $\tau$, also called an intersection *type*.

See Figure 4.3 for a list of all 1-flags on two vertices. The set of all admissible



$S_1^1$ $S_2^1$ $S_3^1$ $S_4^1$

Figure 4.3: If $\tau = 1$ (as permutation), then there are four distinct $\tau$-flags of length two. The empty circle marks $\tau$ in each flag.

$\tau$ flags of length $m$ is denoted by $\mathcal{P}_m^\tau$. If $\tau$ is the permutation of length 0 or 1, we write $\mathcal{P}_m^0$ and $\mathcal{P}_m^1$, respectively. Notice that $\mathcal{P}_m^0 = \mathcal{P}_m$. The *support* $T$ of $\tau$ in $S^\tau$ is the set of indices of $S$ that span $\tau$ in $S^\tau$. We say that two permutation flags $S_1^{\tau_1}$ and $S_2^{\tau_2}$ are *type*-isomorphic if $S_1 = S_2$ and if the supports of $\tau_1$ and $\tau_2$ are identical. For instance, in Figure 4.3, $S_1^1$ and $S_4^1$ are not type-isomorphic, because the support of $\tau$ in $S_1^1$ is 1 and in $S_4^1$ it is 2. For convenience, we set $t := |\tau|$.

**Definition 4.2.3.** Let $S^\tau$ be a $\tau$-flag of length $m$, $P^\tau$ a $\tau$-flag of length $n \geq m$. We define $\#(S^\tau, P^\tau)$ to be the number of $m$-sets $M \subseteq [n]$ such that $P[M]$ is type-isomorphic to $S^\tau$. Flag density is then defined as follows

$$p(S^\tau, P^\tau) = \frac{\#(S^\tau, P^\tau)}{\binom{n-t}{m-t}}.$$

In other words, $p(S^\tau, P^\tau)$ is the probability that a uniformly at random chosen subpermutation of length $m$ from $P^\tau$, subject to it containing $\tau$, induces a a flag type-isomorphic to $S^\tau$. For instance, consider the following flag densities. The empty circle denotes $\tau = 1$.

$$p(\circ\!\bullet, \circ\!\bullet^{\bullet}) = 1, \quad p(\bullet\!\circ, \circ\!\bullet^{\bullet}) = 0, \quad p(\bullet\!\circ, \bullet\!\circ^{\bullet}) = 1/2$$

Finally, we define joint density of two flags, $p(S_1^\tau, S_2^\tau; P^\tau)$, as the probability that choosing an $m_1$-set $M_1 \subseteq [n]$ such that $P[M_1]$ contains $\tau$ and choosing an $m_2$-set $M_2 \subseteq [n]$ such that $P[M_2]$ contains $\tau$ and $M_1 \cap M_2 = \tau$ induces $\tau$-flags $P[M_1]^\tau$ and $P[M_2]^\tau$ in $P^\tau$ which are type-isomorphic to $S_1^\tau$ and $S_2^\tau$, respectively. The following proposition turns out to be useful (Lemma 2.3 in [Raz07]). It says that choosing subflags with or without replacement makes no difference asymptotically.

**Proposition 4.2.1.** *Let $S_1^\tau$ and $S_2^\tau$ be flags on $m_1$ and $m_2$ vertices. Let $n \geq$*

$m_1 + m_2 - t$ *and* $P^\tau$ *be a flag on* $n$ *vertices. Then*

$$p(S_1^\tau, P^\tau)p(S_2^\tau, P^\tau) = p(S_1^\tau, S_2^\tau; P^\tau) + o(1),$$

*where* $o(1) \to 0$ *as* $n \to \infty$.

Let $\ell = |\mathcal{P}_m^\tau|$ and fix an order on elements of $\mathcal{P}_m^\tau$. Let $S_i^\tau, S_j^\tau$ be $\tau$-flags from $\mathcal{P}_m^\tau$ and $P$ a $\tau$-flag from $\mathcal{P}_n^\tau$. Furthermore, let $\mathbf{x}$ be a vector with $i$-th entry $p(S_i^\tau, P^\tau)$, and let $Q^\tau$ be a positive semi-definite matrix with dimensions $\ell \times \ell$. Then by Proposition 4.2.1 and since $Q^\tau \succeq 0$, we have

$$0 \leq \mathbf{x}Q^\tau\mathbf{x}^T = \sum_{i,j\leq\ell} Q_{ij}^\tau p(S_i^\tau, S_j^\tau, P^\tau) + o(1).$$

Moreover, if we let $\sigma$ be a uniformly at random chosen type in $P$ of length $t$, the inequality above remains true. Moreover, an "average" $\sigma$ preserves the non-negativity as well.

$$0 \leq \mathbb{E}_\sigma\left(\mathbf{x}Q^\tau\mathbf{x}^T\right) = \sum_{i,j\leq\ell} Q_{ij}^\tau \frac{1}{\binom{n}{t}} \sum_{\sigma\in\binom{[n]}{t}} p(S_i^\tau, S_j^\tau; P^\sigma) + o(1). \tag{4.3}$$

Next, we write the above expression in terms of permutations on $N$ vertices. Having all information in terms of the same objects allows us to combine it together.

$$\begin{aligned}\mathbb{E}_\sigma\left(\mathbf{x}Q^\tau\mathbf{x}^T\right) &= \sum_{i,j\leq\ell} Q_{ij}^\tau \frac{1}{\binom{n}{t}} \sum_{\sigma\in\binom{[n]}{t}} \sum_{P'\in\mathcal{P}_N} p(S_i^\tau, S_j^\tau; (P', \sigma))p(P', P) + o(1) \\ &= \sum_{P'\in\mathcal{P}_N} \underbrace{\left(\sum_{i,j\leq\ell} Q_{ij}^\tau \frac{1}{\binom{n}{t}} \sum_{\sigma\in\binom{[n]}{t}} p(S_i^\tau, S_j^\tau; (P', \sigma))\right)}_{\alpha(P', m, \tau)} p(P', P) + o(1)\end{aligned}$$

Notice that the last expression is of the form $\sum_{P'\in\mathcal{P}_N} \alpha(P', m, \tau)p(P', P)$. There is one of those for each type $\tau$ and value $m$. Every such choice will require another matrix $Q^\tau$. In practice, we first choose $N$, then take all possible pairs of $t$ and $m$ such that $N = 2m - t$. Thus once $N$ is fixed, the choice of $t$ determines the

rest. Therefore, let $\alpha(P') = \sum_\tau \alpha(P', m, \tau)$ and recall that the expression that we are trying to minimise, subject to $Q^\tau \succeq 0$ for all $\tau$, comes from (4.2). By adding inequalities of the form of (4.3) to (4.2), we obtain

$$\begin{aligned} p(S,P) &= \sum_{P' \in \mathcal{P}_N} p(S,P')p(P',P) \\ &\leq \sum_{P' \in \mathcal{P}_N} p(S,P')p(P',P) + \sum_{P' \in \mathcal{P}_N} \alpha(P')p(P',P) \\ &\leq \max_{P' \in \mathcal{P}_N} \{p(S,P') + \alpha(P')\}. \end{aligned} \tag{4.4}$$

This problem (4.4) has the form of a semidefinite programming problem subject to the condition that $Q^\tau \succeq 0$ for every type $\tau$. There exist numerical solvers, such as CSDP or SDPA, that we can use. However, the solution is in the form of numerical PSD matrices. These need to be converted to exact matrices without floating-point entries in a way that preserves their PSD property and still yields a bound that we are satisfied with. Since none of our bounds is tight, we will take a shortcut in rounding. Let $Q'$ be a numerical matrix returned by the solver. Since it is positive semi-definite, it admits a Cholesky decomposition into a lower and upper triangular matrices: $Q' = L'L'^T$. We compute this decomposition and then round the $L'$ matrices into $L$ matrices in such a way that they do not have negative entries on the diagonals. In certificates, we provide these $L$ matrices instead of $Q$ matrices. This way, one can readily check that $Q = LL^T \succeq 0$ by inspecting the diagonal entries of the $L$ matrices.

### 4.2.2 Example

The following example is a done-by-hand flag algebras method on a small problem of determining the packing density of 132. We have a lower bound of $2\sqrt{3} - 3 \approx 0.464101615\ldots$ given by the standard construction. Assume we want to obtain an upper bound for the packing density of 132. Let $P$ be a (large) 132-maximiser of length $n$ and let $3 \leq \ell \leq n$. By (4.2) we get

$$p(132) \leq p(132, P)$$

$$= \sum_{P' \in \mathcal{P}_\ell} p(132, P')p(P', P)$$
$$\leq \max_{P' \in \mathcal{P}_\ell} p(132, P').$$

We choose $\ell = 3$ and set $\lambda = 2\sqrt{3} - 3$. Now consider

$$\Delta = \lambda p(123, P) + (\lambda - 1)p(132, P) + \lambda p(213, P) + \frac{5\lambda - 3}{6} p(231, P)$$
$$+ \frac{5\lambda - 3}{6} p(312, P) + \lambda p(321, P).$$

Adding the linear combination $\Delta$ of $P'$ densities to the previous crude upper bound improves it to $\lambda$.

$$p(132, P) \leq \sum_{P' \in \mathcal{P}_\ell} p(132, P')p(P', P) + \Delta$$
$$\leq \max_{P' \in \mathcal{P}_\ell} \{\lambda, \lambda, \lambda, \frac{5\lambda - 3}{6}, \frac{5\lambda - 3}{6}, \lambda\}$$
$$= \lambda$$

The key property of $\Delta$ is that it is non-negative for all $P$, including all $P' \in \mathcal{P}_3$. Let $\sigma$ be a randomly chosen vertex out of the three available. The matrix $Q$ below is positive semi-definite and $\mathbf{x}_{P'}$ is a vector of flag densities for flags in Figure 4.3:

$$\mathbf{x}_{P'} = \begin{pmatrix} p(\circ\bullet, (P', \sigma)) & p(\bullet\circ, (P', \sigma)) & p(\circ\bullet, (P', \sigma)) & p(\bullet\circ, (P', \sigma)) \end{pmatrix}.$$

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lambda & \lambda & 3(\lambda - 1)/2 \\ 0 & \lambda & \lambda & 3(\lambda - 1)/2 \\ 0 & 3(\lambda - 1)/2 & 3(\lambda - 1)/2 & 3\lambda \end{pmatrix} \tag{4.5}$$

Averaging over $\sigma$ gives the expression (4.6) that makes the non-negativity of $\Delta$ apparent.

$$\Delta = \mathbb{E}_\sigma \left( \sum_{P' \in \mathcal{P}_3} \mathbf{x}_{P'} Q \mathbf{x}_{P'}^T \right) \geq 0 \tag{4.6}$$

Therefore, we proved that $p(132) \leq 2\sqrt{3} - 3$.

### 4.2.3 Implementation

Flagmatic 2.0 was written by Emil R. Vaughan and is currently the only general implementation of Razborov's flag algebra framework which is freely available to use and modify. See [Vau13] for more information. The project is hosted at `http://github.com/jsliacan/flagmatic`. Unfortunatelly, Flagmatic does not support permutations. For this reason, we wrote Permpack, a lightweight implementation of flag algebras on top of SageMath's Sage 7.4 (see [Dev17]). It does not have all the functionality of Flagmatic but it is sufficient for basic tasks. For more information, code, and installation instructions, see `https://github.com/jsliacan/permpack`.

Let us consider an example of how Permpack can be used on the above example of 132-packing. It will be clear from Permpack's output where the $Q$ matrix above comes from. In Permpack, one needs to specify the complexity in terms of $N$, the length of the admissible permutations which all computations are expressed in terms of. The `density_pattern` argument specifies the permutation whose packing density we want to determine. Once permutations, types, flags, and flag products are computed, we can delegate the rest of the tasks to the solver of our choice (currently supported solvers are `csdp` and `sdpa_dd`). The answer is a numerical upper bound on $p(132)$. It can be rounded automatically to a rational bound by the `exactify()` method of the `PermProblem` class. The certificate contains admissible permutations, flags, types, matrices $Q$ (as $L$ matrices in the Cholesky decomposition of $Q$) and the actual bound as a rational number (fraction). These are suficient to verify the bound. Below is the script used to obtain the numerical $Q'$ matrix for the packing density of 132 with Permpack.

Listing 4.1: Packing 132 with Permpack.

```
p = PermProblem(3, density_pattern="132")
p.solve_sdp()
```

Listing 4.2: Output.

```
...
Success: SDP solved
Primal objective value: -4.6410162e-01
Dual objective value: -4.6410162e-01
Relative primal infeasibility: 5.90e-14
Relative dual infeasibility: 1.67e-10
Real Relative Gap: 3.68e-10
XZ Relative Gap: 6.14e-10
```

It is not difficult to guess the entries of $Q$ from the numerical matrix below, which is part of the output of the SDP solver. The resulting exact matrix $Q$ is shown in (4.5).

Listing 4.3: Floating point $Q'$ matrix.

```
  [ 4.55854035127455e-10  6.806084489120e-12  6.8060845047452e-12 -1.032045390820e-10]
  [ 6.80608448912001e-12  0.4641016162301893  0.464101613919741
-0.8038475767936814]
  [ 6.80608450474521e-12  0.464101613919741   0.4641016162301782
-0.8038475767936717]
  [-1.03204539082084e-10 -0.803847576793681  -0.8038475767936717
1.3923048450288649]
```

## 4.3 Results

The following theorem will be needed later. There exist further variations of it, e.g. Proposition 2.1 and Theorem 2.2 in [AAH+02]. However, we only need the original version.

**Theorem 4.3.1** ([Str93]). *Let $S$ be a layered permutation. Then for every $n$, extremal value of $p(S, n)$ is achieved by a layered permutation. Moreover, if $S$ has no layer of size 1, every maximiser of $p(S, n)$ is layered.*

The scripts used to obtain results in this section can be found at

https://github.com/jsliacan/permpack/tree/master/scripts.

The certificates in support of the upper bounds in this section can be found at the address below. With each result, we provide the name of the certificate file that witnesses it, e.g. `cert1324.js` witnesses the upper bound for $p(1324)$.

https://github.com/jsliacan/permpack/tree/master/certificates.

### 4.3.1 Packing 1324

Layered permutations have been studied in depth by [Pri97]. He came up with an approximation algorithm that, at $m$-th iteration assumes that the extremal construction has $m$ layers (see Theorem 4.3.1) and optimises over their sizes. The algorithm then proceeds to increase $m$ and halts when increasing $m$ does not improve the estimate. In that case, an optimal construction has been found (up to numerical noise from the optimization, if any). In reality, the procedure is stopped manually when approximation is fine enough or the problem becomes too large. Therefore, for every $m$, the value that Price's algorithm gives is a lower-bound for the packing density in question.

It is known that the extremal construction for the packing density of 1324 is layered with infinite number of layers. See, for instance, [AAH+02] and [Pri97]. The main theorem of this section is the following.

**Theorem 4.3.2.**

$$0.244054321 < p(1324) < 0.244054549$$

*Proof.* Consider the construction $\Gamma$ from Figure 4.4, where $\Gamma$ is a permuton. Let $C$ denote the middle layer of $\Gamma$ (the largest layer), $B$ denote the layer above (and $B'$ the layer below) $C$, and $A$ denotes the group of the remaining layers above $B$ (and $A'$ denotes the groupp of layers below $B'$). So $\Gamma = A' \oplus B' \oplus C \oplus B \oplus A$, where $A \oplus B$ means that the layer $A$ is entirely below and to the left of the layer $B$. Let $c = |C|$, $b = |B| = |B'|$, and $a = |A| = |A'|$. We assume that $A$ (and $A'$) is

isomorphic to a maximiser for the packing of 132-pattern (213-pattern). The aim is to optimise over $a$ and $b$. Ideally, the tails of $\Gamma$ would also be optimised over, but that is infeasible. So we assume the tails are 132 (213) maximisers. It turns out that the first two steps give a good lower bound. We now compute the density of 1324 patterns in $\Gamma$. There are four distinct (i.e. up to symmetry) positions that a copy of 1324 can assume in $\Gamma$. Let $xyzw$ be the four points in $\Gamma$ that form a copy of 1324 in that order.

1. $y, z \in C$, $x \in A' \cup B'$, $w \in A \cup B$, there are $N_1$ such copies
2. $y, z \in B$, $x \in A' \cup B' \cup C$, $w \in A$, there are $N_2$ such copies
3. $y, z, w \in A$, $x \in A' \cup B' \cup C \cup B$, there are $N_3$ such copies
4. $x, y, z, w \in A$, , there are $N_4$ such copies

Let us now determine quantities $N_1, \ldots, N_4$.

1. $N_1 = c^2/2 + (a+b)^2$
2. $N_2 = b^2/2 + a(a+b+c)$
3. $N_3 = (2\sqrt{3}-3)\frac{a^3}{6} \cdot (a+2b+c)$
4. $N_4 = \sum_{k=0}^{\infty} \frac{\sqrt{3}\cdot(2\sqrt{3}-3)}{6\cdot(\sqrt{3}+1)^{4k+4}} \cdot a^4$.

Finally, we get the density of 1324 pattern in $\Gamma$. Let $b = 1 - c - 2a$. Then

$$\begin{aligned} p(1324, \Gamma) &= \max_{\substack{0<c\leq 1/2 \\ 0<a<\leq 1/4}} 24 \cdot (N_1 + 2N_2 + 2N_3 + 2N_4) \\ &> 0.244054321. \end{aligned}$$

This proves the lower bound in Theorem 4.3.2, because $0.244054321 < p(1324, \Gamma) \leq p(1324)$.

We use Flagmatic to prove the upper bound. Since 1324 is layered, there is a 1324-maximiser that is layered as well. Therefore, we can limit the search space to the layered permutations. Since Flagmatic does not work with permutations, we transformed the problem to an equivalent problem in directed graphs – which Flagmatic can handle.

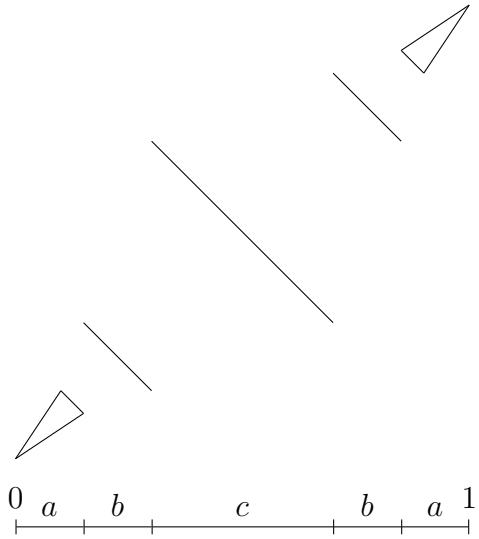
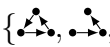
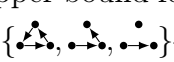



Figure 4.4: Permuton $\Gamma$ provides a lower bound for $p(1324)$. The triangles at the ends represent permutons that are maximisers for the packing of 132 and 213 (L to R).

**Lemma 4.3.1.** *Let $\mathcal{F} =$*  *be the set of forbidden digraphs. The packing density of 1324 equals the Turán*  *-density of $\mathcal{F}$. In other words,*

$$p(1324) = p(\text{  }, \mathcal{F}).$$

*4.3.1.* There is a unique way to encode a layered permutation $P$ as a directed graph $D$. If and only if two points $x, y \in P$ form a 12 pattern, then $xy$ is an arc $x \to y$ in $D$. Forbidding  ,  , and  in $D$ forces it to be a union of independent sets with arcs between them so that if $x, y$ are vertices in one independent set and $u, v$ are vertices in another independent set of $D$, then if $xu$ is an arc in $D$, so are $xv$, $yu$, and $yv$. In other words, all arcs between two independent sets are present, and all go in the same direction. Moreover, the direction is transitive (  is forbidden). Together with the first rule about the direction of arcs between independent sets, this fully characterizes the digraph $D$ from the permutation $P$. Clearly, the process is reversible. □

Given Lemma 4.3.1, we use flag algebra method on directed graphs to compute an upper bound for the packing density of  (an equivalent of 1324 in digraphs) over {  }-free digraphs. The resulting bound is the one in Theorem 4.3.2.

- Certificate: `cert1324flagmatic.js`
- Script: `pack1324flagmatic.sage`

Note that this is a Flagmatic certificate and can be verified using the `inspect_certificate.py` script that comes with Flagmatic.

□

A similar bound can be achieved by Permpack. In particular, we can show that $p(1324) < 0.244054540$.

- Certificate: `cert1324permpack.js`
- Script: `pack1324permpack.sage`

Despite Permpack being able to prove a good bound, we used Flagmatic in the proof above to emphasise that this result had been available before Permpack was written.

### 4.3.2 Packing 1342

The previous lower bound for the packing density of 1342 was approximatelly 0.1965796. The result of [Bat] can be found in [AAH+02].

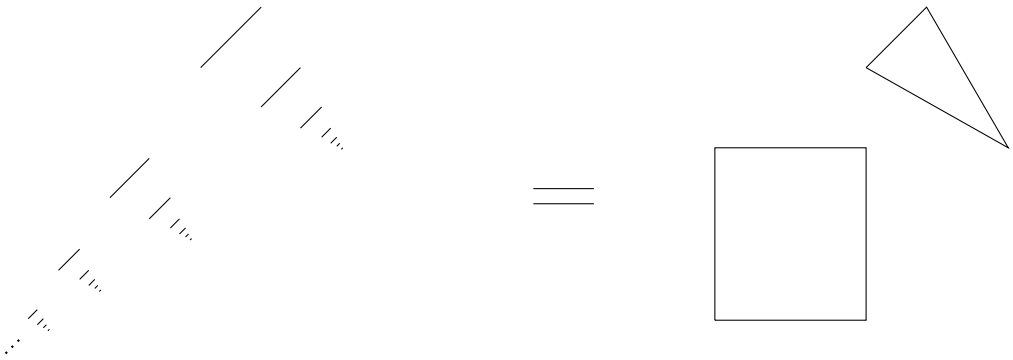


Figure 4.5: On the left is Batkeyev's construction for the lower bound on $p(1342)$ as product of packing densities of 132 and 1432. On the right is the schematic drawing of it. The triangle stands for a 231-maximiser and the square stands for the part inside which the entire construction is iterated.

Let $\lambda = 2\sqrt{3} - 3$ be the packing density of 231 and $\kappa$ the ratio between the top layer and the rest of the 1432-maximiser, see [Pri97] ($\kappa$ is the root of $3x^4 - 4x + 1$). Batkeyev suggested to replace each layer in the maximiser of 1432 by a 231-maximiser while preserving the size ratio $\kappa$. The density of 1342 in Batkeyev's construction (see Figure 4.5) is

$$\begin{aligned} p(1342, B) &= (8\sqrt{3} - 12) \cdot \sum_{n=0}^{\infty} (1-\kappa)^3 \kappa^{4n+1} \\ &= p(132)p(1432) \\ &= 2\left(2\sqrt{3} - 3\right)\left(3\sqrt[3]{\sqrt{2} - 1} - \frac{3}{\sqrt[3]{\sqrt{2} - 1}} + 2\right) \\ &\approx 0.1965796\ldots \end{aligned}$$

This lower bound was widely regarded as possibly optimal. Our contribution to this problem is finding a vastly better lower bound construction. However, if we restrict the space of admissible permutations to those that avoid 2431, then Batkeyev's construction is likely optimal. We are able to prove the following theorem on $N = 6$ admissible graphs to keep the SDP small (if $N = 7$ was chosen, the bound would likely be slightly better).

**Theorem 4.3.3.**

$$p(1342, \{2431\}) < 0.19658177.$$

*Proof.* The materials to verify the theorem are

- Certificate: `cert1342_forb2431.js`
- Script: `pack1342_forb2431.sage`.

□

The following result addresses the actual packing density of 1342 without any forbidden patterns.

**Theorem 4.3.4.**

$$0.198836597 < p(1342) < 0.198837287.$$

*Proof.* The new lower bound is given by the construction Π in Figure 4.6. The weights we used for the parts are given in `cert1342lb.txt`, located with other certificates.

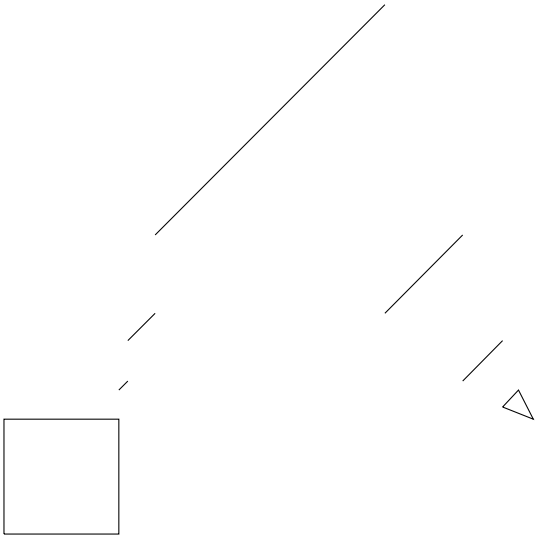


Figure 4.6: New lower bound construction Π for the packing density of 1342. The part sizes, left to right, are approximately 0.2174, 0.0170, 0.0516, 0.4341, 0.1480, 0.0764, 0.0554. The square part represents the part inside which the whole construction is iterated. The triangle part is the extremal construction for 231-packing.

$$\begin{aligned}
a_1 &= 0.21741277235363473086924448 43 \\
a_2 &= 0.01705980578992427227406205 49 \\
a_3 &= 0.05161014024878922702302309 72 \\
a_4 &= 0.43407228098738649943129530 07 \\
a_5 &= 0.14798956259503904962506118 29 \\
a_6 &= 0.07644572558056569713833513 65 \\
a_7 &= 0.05540971244466052363897874 33
\end{aligned} \tag{4.7}$$

Label the 7 parts of Π from left to right as $a_1, \ldots, a_7$. We assign the weights to them roughly as in (4.7). Then a straightforward calculation of the 1342 density in Π implies the desired lower bound. The sage script that does this is called `lb1342.sage`, located with other scripts. For the upper bound, we have

- Certificate `cert1342.js`

- Script is `pack1342.sage`.

□

The upper bound obtained without flag algebras stands at 2/9, see [AAH+02]. The upper bound above was obtained via the flag algebras method and confirms the claimed bound from [BHL+15]. We used $N = 7$ for our computations. While it is possible that $N = 8$ would yield a slightly better bound, the computations would be much more expensive. Without a candidate for an exact lower bound, we were satisfied with the bound we obtained with $N = 7$.

### 4.3.3 Packing 2413

The case packing 2413 patterns is fairly complicated as can be seen from the lower bound construction by [PS10]. The previous upper bound obtained without flag algebras was 2/9 by [AAH+02]. The bound below was obtained via flag algebras and is in the same range as the bound in [BHL+15].

**Theorem 4.3.5.**

$$p(2413) < 0.1047804635435352376177 9.$$

*Proof.* The materials to verify the theorem are

- Certificate: `cert2413.js`
- Script: `pack2413.sage`.

□

We used admissible permutations of length $N = 7$. Again, larger $N$ could yield a slightly better upper bound, but without an exact lower bound, this effort would not be justified.

## 4.4 Packing other small permutations

The flag algebras method will yield upper bounds for many problems. In some cases these bounds are particularly interesting because they are close to their corresponding lower bounds. In this section we list a selection of upper and lower bounds that are potentially sharp since their values appear to be close to each other.

In the list below we choose to represent the permutations by their drawings in the grid. This is more transparent as the permutations became larger. The extremal constructions (permutons) on the left-hand side of the Table 4.2 are represented by their drawings as well. The lower bounds are given on the left-hand side of the table and upper bounds on the right-hand side of the table. This is a sample of the results obtained with Permpack via flag algebras.

We now give the descriptions of the lower bound constructions.

1. For 23154 =  , the construction is a sum of two parts in ratio 2 : 3 top to bottom. The bottom part is a 231-maximiser while the top part is a simple decreasing segment.

   - Certificate: `cert23154.js`
   - Script: `pack23154.sage`

2. For 14523 =  , the construction is designed as follows. Let $\alpha$ be the maximiser of $5(1-x)^4/(1-x^5)$ such that $\alpha \in [0,1]$. The topmost sum-indecomposable part of the  -maximiser has length $\alpha$ and the remainder of the maximiser has length $(1-\alpha)$. The construction is iterated inside the part of length $(1-\alpha)$. The part of length $\alpha$ is a skew-sum of two balanced increasing segments. The exact value of the density on the left-hand side of Table 4.2 is too complicated to fit in.

   - Certificate: `cert14523.js`
   - Script: `pack14523.sage`

3. For 21354 =  , the construction is a 4-layered permuton with layers of lengths $\beta, 1/2-\beta, 1/2-\beta, \beta$, top to bottom. Here, $\beta$ is the real root of

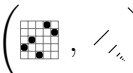
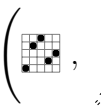
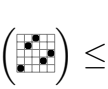
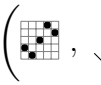
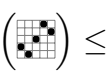
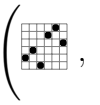
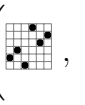
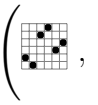
| | $\sigma$ | bounds |
|---|---|---|
| 1. | 23154 | $5!\frac{(2/5)^2}{2!}\frac{(3/5)^3}{3!}(2\sqrt{3}-3) = p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.16039\ldots$ |
| 2. | 14523 | $0.153649\ldots \sim p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.153649\ldots$ |
| 3. | 21354 | $0.16515\ldots \sim p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.16515\ldots$ |
| 4. | 231654 | $6!\frac{(1/2)^6}{3!^2}(2\sqrt{3}-3) = p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.145031\ldots$ |
| 5. | 231564 | $(2\sqrt{3}-3)^2\frac{6!}{48^2} = p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.0673094$ |
| 6. | 231645 | $(2\sqrt{3}-3)^2\frac{6!}{48^2} = p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.0673094$ |
| 7. | 215634 | $\frac{6!}{9^3 2^3} = p\left(\right.$  $\left.\right) \le p\left(\right.$  $\left.\right) \le 0.123456\ldots$ |

Table 4.2: Exact values are known for all densities on the left-hand side. They are described in the text as they are not easy to write down.

$40x^3 - 32x^2 + 9x - 1 = 0$. Again, we only write the approximate value on the left-hand side in Table 4.2 for space reasons.

- Certificate: `cert21354.js`
- Script: `pack21354.sage`

4. For 231654 =  , the construction is identical in structure to the construction for  , except the ratios of the two parts in the sum are $1:1$.

   - Certificate: `cert231654.js`
   - Script: `pack231654.sage`

5. For 231564 =  , the construction is the sum of two 231-maximisers of equal size.
   - Certificate: `cert231564.js`
   - Script: `pack231564.js`
6. For 231645 =  , the construction resembles that of 231564 except the top 231-maximiser is flipped accordingly.
   - `cert231645.js`
   - Scripts `pack231645.js`
7. For 215634 =  , the construction has three segments of equal length arranged as portrayed in Table 4.2.
   - Certificate: `cert215634.js`
   - Script: `pack215634.sage`

## 4.5 Conclusion

While we now know the packing densities of all 4-point permutations with accuracy of 0.01%, finding candidates for optimal constructions for the cases of 1324 and 1342 remains a challenge. In the case of 1324, a new idea for the part ratios will be needed to come up with a possible extremal construction. As for the 1342 pattern, the extremal construction might use a different layer formation than our $\Pi$. Even if $\Pi$ has the right structure, the part ratios remain to be determined precisely. The latest status of 4-point packing densities is depicted in Table 4.3.

After 4-point permutations, there are many packing densities of small permutations of length $5, 6, \ldots$. The values of lower bounds and upper bounds in Table 4.2 should be made to match. In some cases this will be easier than in others. In particular, the packing density of 21354 has been mentioned in both [AAH+02] and [Häs02].

There are analogous questions to be asked about packing densities when certain patterns are forbidden. As an example, we mentioned $p(1342, \{2431\})$ in relation to $p(1342)$.

| S | lower bound | ref LB | upper bound | ref UB |
|---|---|---|---|---|
| 1234 | 1 | trivial | 1 | trivial |
| 1432 | $\beta$ | [Pri97] | $\beta$ | [Pri97] |
| 2143 | 3/8 | trivial | 3/8 | [Pri97] |
| 1243 | 3/8 | trivial | 3/8 | [AAH+02] |
| 1324 | 0.244054321* | – | 0.244054549* | – |
| 1342 | 0.198836597* | – | 0.198837286342* | – |
| 2413 | $\approx$ 0.104724 | [PS10] | 0.104780463544* | – |

Table 4.3: Overview of packing densities for 4-point permutations given the information in this paper. The values with asterisk have been updated.

Next, an interesting line of enquiry was made precise as Conjecture 9 in [AAH+02]. For a packing of pattern $S$, is there an extremal construction with infinite number of layers? Are all extremal constructions of that form? More precisely, let an *$S$-maximiser* be an $n$-permutation $P$ such that $p(S, n) = p(S, P)$. If $L_n$ is the number of layers in a layered maximiser of length $n$, what can we say about $L_n$ as $n \to \infty$? For example, we know that the number of layers in every 1324-maximiser is unbounded as $n \to \infty$. We also know that a 2143-maximiser has only two layers, regardless of $n$.

# Chapter 5

# Permpack

In Chapter 4, we used the flag algebras method when bounding packing densities of small permutations. The method transforms a combinatorial problem to a semidefinite one. See Section 4.2.1 for details about the flag algebras framework. Since flag algebras are algorithmic and syntax-based, they are suitable for a computer implementation.

Permpack [Sli16] is a lightweight implementation of flag algebras as a package on top of Sage [Dev17]. Permpack is exclusively for flag algebras computations on permutations. The syntax is supposed to resemble that of Flagmatic as much as possible, where Flagmatic is a flag algebras software for graphs, 3-uniform hypergraphs, directed graphs, and multigraphs developed by Vaughan [Vau13]. A version of Flagmatic in its original form with minimal updates (by me) to maintain compatibility with new versions of Sage is available from `https://github.com/jsliacan/flagmatic`. Similarly, the source code of Permpack can be found in its Github repository `https://github.com/jsliacan/permpack`.

## 5.1 Set-up

To use Permpack, you will need a UNIX machine (a flavour of Linux or a Mac) with a recent (7.5+) Sage installed on it (note that Sage has its own list of dependencies). Additionally, one also needs a semidefinite solver to which Permpack passes the constructed semidefinite problem.

Installing Sage from source is recommended if speed is an issue, which it is in most interesting cases. This can take up to a day on a regular consumer laptop! Instructions can be found here: http://doc.sagemath.org/html/en/installation/source.html. It is useful to specify the number of jobs to run in parallel when making Sage. This can be done with the following command: `$ MAKE='make -jNUM' make`, where `NUM` is the number of jobs you can afford simultaneously. Once Sage is up and running, you will need to place the solver's binary file in the `$PATH`, so that Sage can find it when needed. Usually `/user/local/bin` works. At this point, the set-up should be complete.

### 5.1.1 Solvers

There are two semidefinite solvers that are currently supported by Permpack: CSDP [Bor99] and SDPA-DD [YFF+12]. It is advisable to install CSDP from source (including LAPACK/BLAS libraries) in order to make the best use of the resources on the computer that you are using. See the instructions in the INSTALL file included with the source distribution on Github: https://github.com/coin-or/Csdp.

Among other available solvers are SeDuMi, SDPT3, DSDP, and CVXOPT. These are all based on the interior-point method. First order method is at the core of SCS solver. In practice, CSDP and various precisions of SDPA solver (SDPA-DD, SDPA-QD) perform reasonably well both in terms of accuracy and speed. Permpack supports CSDP (`csdp`) and SDPA-DD (`sdpa_dd`) options for the `solver` argument.

**Note.** When using SDPA double precision solver, the numerical solution matrices that the solver returns are not always positive semidefinite. There may be eigenvalues which are negligibly negative, say $-10^{-20}$. This needs to be treated before further processing because we use numpy's Cholesky decomposition in the next step. Hence, Permpack cannot currently deal with negative eigenvalues, however small.

## 5.2 Usage

It is ideal to start Permpack from `/path/to/permpack/pkg/` directory. Once there, type `sage`. This should start Sage and you should see something similar to Listing 5.1.

Listing 5.1: Starting Sage.

```
------------------------------------------------------------------
| Sage Version 8.0, Release Date: 2017-07-21                     |
| Type "notebook()" for the browser-based notebook interface.    |
| Type "help()" for help.                                        |
------------------------------------------------------------------
sage:
```

You can test that things were loaded and that Sage can see your SDP solver by running a small example as in Listing 5.2.

Listing 5.2: Floating point upper bound on the packing density of 132.

```
1 sage: from permpack.all import *
2 sage: p = PermProblem(3, density_pattern="132")
3 sage: p.solve_sdp(solver="csdp")
```

The output should resemble the one in Listing 5.3, with the bound being reported as `-4.6410162e-01` (the minus sign is due to the internal representation of the SDP problem in CSDP as maximization versus minimization). Clearly, the result is the floating point representation of $2\sqrt{3} - 3$, the packing density of 132.

Listing 5.3: Output from Permpack after commands from Listing 5.2.

```
1  Generating admissible permutations... OK.
2  Generating types... OK.
3  Generating flags... OK.
4  Expressing density pattern as a linear combination of permutations... OK.
5  ------------------------------------------------
6  Generated:
7  6 admissible permutations.
8  1 types of order 1, with [4] flags.
9  ------------------------------------------------
10 /path/to/Github/permpack/pkg/../store/FP-N3.txt
11 Loading flag_products from file... OK.
12 Writing SDP input file... OK.
13 Solving SDP problem...
14 Iter:  0 Ap: 0.00e+00 Pobj: -1.200000e+02 Ad: 0.00e+00 Dobj:  0.000000e+00
15 Iter:  1 Ap: 9.68e-01 Pobj: -2.014663e+02 Ad: 9.36e-01 Dobj:  9.579035e+00
16 Iter:  2 Ap: 1.00e+00 Pobj: -1.569510e+02 Ad: 1.00e+00 Dobj:  9.009622e+00
17 Iter:  3 Ap: 9.92e-01 Pobj: -8.120415e+00 Ad: 1.00e+00 Dobj:  5.149692e-01
18 Iter:  4 Ap: 9.98e-01 Pobj: -1.207828e+00 Ad: 1.00e+00 Dobj: -3.893852e-02
```

```
Iter:  5 Ap: 1.00e+00 Pobj: -9.312314e-01 Ad: 7.51e-01 Dobj: -3.164694e-01
Iter:  6 Ap: 9.92e-01 Pobj: -5.212294e-01 Ad: 1.00e+00 Dobj: -4.256196e-01
Iter:  7 Ap: 1.00e+00 Pobj: -4.689616e-01 Ad: 1.00e+00 Dobj: -4.609830e-01
Iter:  8 Ap: 9.99e-01 Pobj: -4.643160e-01 Ad: 1.00e+00 Dobj: -4.639517e-01
Iter:  9 Ap: 9.99e-01 Pobj: -4.641108e-01 Ad: 1.00e+00 Dobj: -4.640964e-01
Iter: 10 Ap: 1.00e+00 Pobj: -4.641020e-01 Ad: 1.00e+00 Dobj: -4.641023e-01
Iter: 11 Ap: 1.00e+00 Pobj: -4.641016e-01 Ad: 1.00e+00 Dobj: -4.641016e-01
Iter: 12 Ap: 9.56e-01 Pobj: -4.641016e-01 Ad: 9.60e-01 Dobj: -4.641016e-01
Success: SDP solved
Primal objective value: -4.6410162e-01
Dual objective value: -4.6410162e-01
Relative primal infeasibility: 4.65e-14
Relative dual infeasibility: 1.67e-10
Real Relative Gap: 3.68e-10
XZ Relative Gap: 6.14e-10
DIMACS error measures: 4.65e-14 0.0e+00 3.10e-10 0.0e+00 3.68e-10 6.14e-10
Finished. OK.
```

Before using any of the module, it is necessary to load it. This is done by the command in Sage interpreter given in Listing 5.4.

Listing 5.4: Loading permpack module.

```
sage: from permpack.all import *
```

### 5.2.1 Entering the problem into Permpack

Assuming that the permpack module is loaded, the user is concerned with only one class: `PermProblem`. Everything is done in terms of solving this permutation problem. It is easily created (and assigned to a variable through which it is accessed later) as in Listing 5.5.

Listing 5.5: Creating a permutation problem to solve.

```
sage: from permpack.all import *
sage: p = PermProblem()
```

The constructor for the `PermProblem` class takes at most three arguments (described in Table 5.1), none of which is mandatory. The `__init__()` method of `PermProblem` assumes that the length of admissible permutations is zero, the density pattern is 21, and the set of forbidden permutations is empty. In other words, the default `PermProblem` object is just a dummy permutation problem. Therefore, one usually wants to specify at least the length of admissible permutations `N` and the *density pattern*.

| Argument | Description |
|---|---|
| `N` | length of each admissible permutation |
| `forbid` | list of forbidden permutations |
| `density_pattern` | linear combination of permutations whose density we maximize |

Table 5.1: Arguments to the `PermProblem` constructor.

For example, when solving the packing problem for 132, it is sufficient to consider `N` $= 3$ (see example in Section 4.2.2). Notice that `N` $< 3$ would not be useful as the density of 132 in any permutation of length less than 3 is zero, hence the density behaviour of 132 would not be captured in, say, admissible permutations of length two: $\mathcal{P}_2 = \{12, 21\}$. Usually, higher `N` yields more precise results (unless the problem is intrinsically hard for flag algebras method, as could be the case when the bound is attained by more than one extremal construction). To keep the section self-contained, we give an example of how to solve the packing problem for 132 via Permpack. See Listing 5.6.

Listing 5.6: Setting up packing problem for the 132 pattern.

```
1 sage: from permpack.all import *
2 sage: p = PermProblem(3, forbid=[], density_pattern="132")
```

The response from Permpack should resemble the output in Listing 5.7.

Listing 5.7: Response from Permpack when setting up the packing problem for 132.

```
1  Generating admissible permutations... OK.
2  Generating types... OK.
3  Generating flags... OK.
4  Expressing density pattern as a linear combination of permutations... OK.
5  ------------------------------------------------
6  Generated:
7  6 admissible permutations.
8  1 types of order 1, with [4] flags.
9  ------------------------------------------------
10 /path/to/Github/permpack/pkg/../store/FP-N3.txt
11 Loading flag_products from file... OK.
```

Notice, in particular, the lines 10-11:

```
/path/to/Github/permpack/pkg/../store/FP-N3.txt
Loading flag_products from file... OK.
```

Permpack does not recompute flag products if the problem (requiring the same flag products) was already encountered before. This is to save time. We briefly mention this in Section 5.3.

Let us now comment on the format of the input to the `PermProblem` constructor.

- `N`: Permpack only accepts integers. When `N` gets too large (say 9), the computations become very slow already when computing flag products. It is very likely that even if you manage to wait long enough to compute these, the SDP problem will be so large that you will run out of memory (for instance).

- `forbid`: Permpack accepts a list of permutations. Do make sure that each of them has length at most `N`. Otherwise it is impossible to exclude them from the computations.

- `density_pattern`: Permpack accepts a few different inputs. Firstly, a solo permutation either as a list of integers, `[1,3,2]`, or as a string,`"132"`, is fine. However, it is possible to want to maximize the density of, say, $1/2 \cdot 132 + 1/2 \cdot 231$. This can also be done. Permpack takes such input as a list of tuples, each touple of length two with the first entry a permutation (either as a list of integers or as a string) and the second entry a coefficient (as a fraction or a floating point number). For example, the following inputs are all equivalent (although the last one is scaled differently – the problem has the same maximizer(s), just different packing density of the `density_pattern`):

  ⋆ `density_pattern=[([1,3,2],1/2),([2,1,3],1/2)]`

  ⋆ `density_pattern=[("132",0.5),("213",0.5)]`

  ⋆ `density_pattern=[("132",1),("213",1)]`

### 5.2.2 Solving SDP

The next method one needs to call on the `PermProblem` is `solve_sdp()`. As usual, it is possible to call this method without any arguments. In such case, the solver defaults to CSDP. If one wants a double precision solver SDPA-DD, argument needs to be passed with that option: `solve_sdp(solver="sdpa_dd")`. First, the method `solve_sdp()` writes and input file to the SDP solver. The default input filename is `sdp.dat-s`. If you wish for a different name pass an additional argument to the `solve_sdp()` method specifying the filename, e.g.

`input_file="myfile.dat-s"`. Make sure that the filename ends with `.dat-s` suffix to be recognized as a sparse data file by the solver. The `solve_sdp()` method then passes the information to the SDP solver together with the output filename, where the solution to the semidefinite problem will be stored. The default output filename is `sdp.out`. If you wish this file to be named differently, you need to pass an additional argument to the `solve_sdp()` method specifying your preferred filename: `output_file="myfile.out"`, where the `.out` extension is just a convention. All these files are stored in the current working directory to be easily accessible if one wants to peruse them manually. Once the method passes everything to the solver, it can take a long time before a solution is found by the solver. You will see the progress on your screen as the output from the solver is not suppressed. Once the solver finished, you'll be able to see the floating point bound that was found. The next step is rounding the floating point solution matrices to have exact entries such that they stay positive semidefinite and the new exact bound is as good as possible. An example call to `solve_sdp()` is in Listing 5.8.

Listing 5.8: Packing 132. Code up to calling the SDP solver. You should see `myfile.dat-s` and `myfile.out` in your current working directory after executing the following code.

```
sage: from permpack.all import *
sage: p = PermProblem(3, density_pattern="132")
sage: p.solve_sdp(solver="csdp",
                  input_file="myfile.dat-s",
                  output_file="myfile.out")
```

### 5.2.3 Assumptions

A feature of Permpack that is worth mentioning, albeit briefly, are assumptions. Imagine one wants to add additional density assumptions on various pattern densities. There are various natural examples, but a small and easily checkable one is the following. Imagine that you want to find the packing density of 12 in 123-free permutations. However, you insist that the density of 312 is at least 1/5 (notice that its density would be 0 in the maximiser of 12 in 123-free permutations). Therefore, you are essentially forcing the problem away from the usual optimum. It is not immediately obvious what the answer is, and the number that Permpack

gives will probably not make you wiser. It was an arbitrarily chosen problem to illustrate a feature. However, you can certainly see that the value Permpack gives you is smaller than 1/2, which would be the packing density of 12 in 123-free permutations. See the code snippet in Listing 5.9 that asks Permpack to compute this example for us. The structure of the command is the following. Use the `add_assumption` method of the `PermProblem` class. It takes two arguments: a list of pairs $(\sigma, c_\sigma)$ and a bound $b$. The idea is to bound a linear combination of permutation densities, such as $c_{123}p(123, P) + c_{321}p(321, P) \geq b$, where the inequality holds for every admissible $P$. Clearly, this can be encoded as a vector of pairs $(\sigma, c_\sigma)$ and an additional bound $b$. In our example, $\sigma = 312$, $c_{312} = 1$, and $b = 1/5$.

Listing 5.9: Forbid 123, maximise 12 under the constraint that $p(312) \geq 1/5$.

```
sage: from permpack.all import *
sage: p = PermProblem(3, forbid=["123"], density_pattern="12")
sage: p.add_assumption([("312",1)], 1/5)
sage: p.solve_sdp()
```

The critical excerpt from the output is provided below, in Listing 5.10.

Listing 5.10: The interesing part of Permpack's output for the problem in Listing 5.9.

```
Success: SDP solved
Primal objective value: -3.4261685e-01
Dual objective value: -3.4261686e-01
Relative primal infeasibility: 9.25e-15
Relative dual infeasibility: 6.65e-09
Real Relative Gap: -1.16e-08
XZ Relative Gap: 7.77e-09
DIMACS error msrs: 9.25e-15 0.00e+00 1.39e-08 0.00e+00 -1.16e-08 7.77e-09
Finished. OK.
```

### 5.2.4 Rounding

The next aspect to discuss is the rounding procedure. As is obvious from above, a SDP solver returns matrices whose entries are floating point values. They are inexact and do not prove anything. Therefore, we need to convert these matrices to other matrices, preserving positive semi-definiteness and ideally stay as close to the desired bound on our problem as possible. We call the procedure of converting floating-point matrices to "exact" matrices, *rounding* and is done by `exactify()` method in Permpack. It is quite basic at this point. We only worry about preserving the positive-semidefiniteness and hope that if the rounded matrix is entry-wise

close to the original one, then the bound will not be off by too much. In practice, this works reasonably well. The method `exactify` takes several optional arguments which can fine-tune the "precision" with thich the rounding is done, e.g. eigenvalues close to zero are rounded to zero if they are close enough (this is the parameter `recognition_precision` passed to `exactify`; another parameter is `rounding_precision` and it specifies the denominator when rounding the entries of $L'$ in the Cholesky decomposition of a floating-point $Q'$ to rational quantities). We give a shared example for rounding and issuing certificates in Subsection 5.2.5, see Listing 5.11 and 5.12 for input and output, respectively.

### 5.2.5 Certificates

The last critical part of Permpack that the user should be aware of are certificates. These are human readable files that store information necessary to verify the results obtained by Permpack. The key idea being that it is significantly easier to check if given matrices solve a SDP problem than finding those matrices in the first place. Therefore, the class `PermProblem` has a method `write_certificate` which takes filename you want your certificate to have. It will be written into the current directory (so make sure you have writing permissions there). The certificate is a JSON file and is easy to read in and process. It contains the following information:

1. admissible permutations
2. flags
3. types
4. matrices $L$
5. bound (rational)

Notice that as things currently stand, rounding (recall `exactify()` method) is done via Cholesky decomposition of the SDP $Q$ matrices. Therefore, every floating-point $Q' = L'L'^T$ is stored as the rounded version of $L'$ from which a rounded version of $Q'$ can be recovered. The example below shows how to round the SDP matrices and issue a certificate for your problem.

Listing 5.11: We want eigenvalues smaller than $10^{-6}$ to be rounded to 0 and the denominator for rounding to be $10^{10}$.

```
sage: from permpack.all import *
sage: p = PermProblem(3, forbid=["123"], density_pattern="12")
sage: p.solve_sdp()
sage: p.exactify(recognition_precision=10e-6, rounding_precision=10e10)
sage: p.write_certificate("babyproblem.cert")
```

Listing 5.12: The interesing part of Permpack's output for the problem in Listing 5.11.

```
...
Success: SDP solved
Primal objective value: -5.0000001e-01
Dual objective value: -5.0000001e-01
Relative primal infeasibility: 9.44e-16
Relative dual infeasibility: 6.32e-09
Real Relative Gap: -9.94e-10
XZ Relative Gap: 6.94e-09
DIMACS error msrs: 9.44e-16 0.00e+00 1.19e-08 0.00e+00 -9.94e-10 6.94e-09
...
Transforming floating-point matrices to rational matrices...
Reading output of the CSDP solver... OK.
Rounding Q matrices... OK.
Computing exact bound...
[OK]   Done. Exact bound is roughly 0.5000000043. Access it from
self._exact_bound.
Writing certificate into file...
[OK]   Certificate written successfully to babyproblem.cert.
```

## 5.3 Miscellaneous

Here are a couple of remarks about Permpack to help you understand mysteriously looking output or behaviour of the package.

First, Permpack stores your computations if they are new. Indeed, when you clone the repository, it comes with some clumsily big files containing flag products for particular problems. Flag products are expensive to compute and at some point it becomes useful to store them and next time only load them from file. We intentionally did not choose any database storage in order for the files to be human readable. They are plain text files.

Second, Permpack does make small use of multiprocessing when computing flag products. Maybe it would be better to use multiple threads instead, but this is something to consider at a later point in time if Permpack ever gets rewritten.

The parallelization offers a modest speed-up if the machine has several cores. No miracles.

## 5.4 Conclusion

There are several hidden features of Permpack that would require longer discussion of the underlying Flag Algebras method. For instance, one can access slack variables of an SDP problem and, provided the bound is tight, obtain information about which admissible permutations do not asymptotically appear in any extremal construction for the problem. Additionally, there are hidden variables starting with an underscore (`_`) that give the user access to intermediate computations and auxiliary variables. These can be useful in certain circumstances. The Flag Algebras method offers various insights into the problem through the computations that are performed on the way to obtaining a sharp bound. For discussion about these, follow the references in Section 4.2.1.

Permpack's functionality could be extended in many directions. From more general assumptions (I have implemented theses in Flagmatic for hypergraphs), through supply of extremal constructions that help with rounding (see the original Flagmatic [Vau13]), to a more complete rounding procedure. Also, it is quite possible that, as in case of graphs, there are automated ways to determine various levels of stability of extremal constructions — see a recent preprint [PST17].

Finally, I would like to invite programmers to take a look at Permpack. There are areas which require no knowledge of Flag Algebras. For instance, at the end, solutions need to be rounded – by this we mean converted from floating-point matrices to either matrices with rational entries or entries from some particular extension of rationals. These computations are not at all trivial, but do not depend on the idea of Flag Algebras. One only needs to understand the format of the SDP (see Section 4.2.1, towards the end). Given an efficient rounding procedure, many problems would be automated from the input line all the way to the certificate stage. A good example of a worthy contribution is by Eric Zhang who simply wrote a more efficient `normalize` function which is used inside loops in critical computations. There must be many such inefficiencies and removing them often requires no knowledge of Flag Algebras.

# Bibliography


[AA05] M. H. Albert and M. D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1):1–15, 2005.

[AAB11] M. H. Albert, M. D. Atkinson, and R. Brignall. The enumeration of permutations avoiding 2143 and 4231. *Pure Mathematics and Applications*, 22:87–98, 2011.

[AAB12] M. H. Albert, M. D. Atkinson, and R. Brignall. The enumeration of three pattern classes using monotone grid classes. *Electronic Journal of Combinatorics*, 19, 06 2012.

[AAB+13] M. H. Albert, M. D. Atkinson, M. Bouvel, N. Ruškuc, and V. Vatter. Geometric grid classes of permutations. *Transactions of the American Mathematical Society*, 365(11):5859–5881, 2013.

[AAH+02] M. H. Albert, M. D. Atkinson, C. C. Handley, D. A. Holton, and W. Stromquist. On packing densities of permutations. *Electronic Journal of Combinatorics*, 9(1), 2002.

[AAV14] M. H. Albert, M. D. Atkinson, and V. Vatter. Inflations of geometric grid classes: three case studies. *Australasian Journal of Combinatorics*, 58(1):27–47, 2014.

[AB14] M. H. Albert and R. Brignall. Enumerating indices of schubert varieties defined by inclusions. *Journal of Combinatorial Theory, Series A*, 123(1):154–168, 2014.

[AB16] M. H. Albert and R. Brignall. $2 \times 2$ monotone grid classes are finitely based. *Discrete Mathematics and Theoretical Computer Science*, 18(2), Permutation Patterns 2015, 2016.

[ABRV16] M. H. Albert, R. Brignall, N. Ruškuc, and V. Vatter. Rationality for subclasses of 321-avoiding permutations. preprint, arXiv:1602.00672, 2016.

[AER+06] M. H. Albert, M. Elder, A. Rechnitzer, P. Westcott, and M. Zabrocki. On the stanley–wilf limit of 4231-avoiding permutations and a conjecture of arratia. *Advances in Applied Mathematics*, 36(2):96 –105, 2006. Special Issue on Pattern Avoiding Permutations.

[Alb10] M. H. Albert. An introduction to structural methods in permutation patterns. In Linton S., Ruškuc N., and Vatter V., editors, *Permutation Patterns, London Mathematical Society Lecture Note Series*, pages 41–66. Cambridge University Press, 2010.

[Alb12] M. H. Albert. PermLab: Software for permutation patterns. http://www.cs.otago.ac.nz/PermLab, 2012.

[Arr99] R. Arratia. On the stanley-wilf conjecture for the number of permutations avoiding a given pattern. *Electronic Journal of Combinatorics*, 6(1), 1999.

[Atk98] M. D. Atkinson. Permutations which are the union of an increasing and a decreasing subsequence. *Electronic Journal of Combinatorics*, 5, 02 1998.

[Atk99] M. D. Atkinson. Restricted permutations. *Discrete Mathematics*, 195(1):27–38, 1999.

[Bat] B. Batkeyev. Extremal construction for 1342-packing. unpublished.

[Bev15a] D. I. Bevan. Growth rates of permutation grid classes, tours on graphs, and the spectral radius. *Transactions of the American Mathematical Society*, 367(8):5863–5889, 2015.

[Bev15b] D. I. Bevan. *On the growth of permutation classes*. PhD thesis, The Open University, 2015.

[Bev15c] D. I. Bevan. Permutation patterns: basic definitions and notation. preprint, arXiv:1506.06673, 2015.

[Bev17] D. I. Bevan. The permutation class av(4213,2143). *Discrete Mathematics and Theoretical Computer Science*, 18(2), 4 2017.

[BHL+15] J. Balogh, P. Hu, B. Lidický, O. Pikhurko, B. Udvari, and J. Volec. Minimum number of monotone subsequences of length 4 in permutations. *Combinatorics, Probability and Computing*, 24(04):658–679, 2015.

[Bon97] J. A. Bondy. Counting subgraphs a new approach to the Caccetta-Häggkvist conjecture. *Discrete Mathematics*, 165:71–80, 1997.

[Bor99] E. B. Borchers. Csdp package. https://projects.coin-or.org/Csdp/, 1999.

[Bri10] R. Brignall. A survey of simple permutations. In Linton S., Ruškuc N., and Vatter V., editors, *Permutation Patterns, London Mathematical Society Lecture Note Series*, pages 41–66. Cambridge University Press, 2010.

[Bri12] R. Brignall. Grid classes and partial well order. *Journal of Combinatorial Theory. Series A*, 119(1):99–116, 2012.

[BS17] R. Brignall and J. Sliačan. Juxtaposing catalan permutation classes with monotone ones. *Electronic Journal of Combinatorics*, 24(2)(2), 2017.

[BT11] R. Baber and J. Talbot. Hypergraphs do jump. *Combinatorics, Probability and Computing*, 20(2):161–171, 2011.

[Bó05] M. Bóna. The limit of a stanley–wilf sequence is not always rational, and layered patterns beat monotone patterns. *Journal of Combinatorial Theory, Series A*, 110(2):223–235, 2005.

[CGW88] F. R. K. Chung, R. L. Graham, and R. M. Wilson. Quasirandom graphs. *Proceedings of the National Academy of Sciences of the United States of America*, 85(4):969–970, 1988.

[Dev17] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.4)*, 2017. http://www.sagemath.org.

[Fox13] J. Fox. Stanley-wilf limits are typically exponential. preprint, arXiv:1310.8378, 2013.

[FRMPV15] V. Falgas-Ravry, E. Marchant, O. Pikhurko, and E. R. Vaughan. The codegree threshold for 3-graphs with independent neighborhoods. *SIAM Journal on Discrete Mathematics*, 29(3):1504–1539, 2015.

[FRV12] V. Falgas-Ravry and E. R. Vaughan. Turán H-densities for 3-graphs. *The Electronic Journal of Combinatorics*, 19(3):P40–, 2012.

[FRV13] V. Falgas-Ravry and E. R. Vaughan. Applications of the semi-definite method to the Turán density problem for 3-graphs. *Combinatorics, Probability and Computing*, 22(01):21–54, 2013.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[Häs02] P. A. Hästö. The packing density of other layered permutations. *Electronic Journal of Combinatorics*, 9(2), 2002.

[HKM+13] C. Hoppen, Y. Kohayakawa, C. G. Moreira, B. Ráth, and Sampaio M. R. Limits of permutation sequences. *Journal of Combinatorial Theory. Series B*, 103(1):93–113, 2013.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 2nd edition, 2001.

[Hua14] H. Huang. On the maximum induced density of directed stars and related problems. *SIAM Journal on Discrete Mathematics*, 28(1):92–98, 2014.

[Inca] The OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. http://oeis.org.

[Incb] Wolfram Research, Inc. Mathematica, Version 10.4. Champaign, IL, 2016.

[KNS64] G. Katona, T. Nemetz, and M. Simonovits. On a problem of turán in the theory of graphs. *Matematikai Lapok*, 15:228–238, 1964.

[KP13] D. Král' and O. Pikhurko. Quasirandom permutations are characterized by 4-point densities. *Geometric and Functional Analysis*, 23(2):570–579, 2013.

[Lov12] L. Lovász. *Large networks and graph limits*, volume 60. American Mathematical Society, 2012.

[LRV10] S. Linton, N. Ruskuc, and V. Vatter. *Permutation Patterns*. Cambridge University Press, New York, NY, USA, 2010.

[M.10] Klazar M. Some general results in combinatorial enumeration. In Linton S., Ruškuc N., and Vatter V., editors, *Permutation Patterns, London Mathematical Society Lecture Note Series*, pages 41–66. Cambridge University Press, 2010.

[Min16] S. Miner. Enumeration of several two-by-four classes. preprint, arXiv:1610.01908, 2016.

[MT04] A. Marcus and G. Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *Journal of Combinatorial Theory. Series A*, 107(1):153–160, 2004.

[MV03] M. M. Murphy and V. Vatter. Profile classes and partial well-order for permutations. *Electronic Journal of Combinatorics*, 9(2), 2003.

[Pan17] J. Pantone. The enumeration of permutations avoiding 3124 and 4312. *Annals of Combinatorics*, 21(2):293–315, 2017.

[Pri97] A. L. Price. *Packing densities of layered* patterns*. PhD thesis, University of Pennsylvania, 1997.

[PS10] C. B. Presutti and W. Stromquist. Packing rates of measures and a conjecture for the packing density of 2413. *Permutation patterns*, 376:287–316, 2010.

[PST17] O. Pikhurko, J. Sliačan, and K. Tyros. Strong forms of stability from flag algebra calculations. preprint, arXiv:1706.02612, 06 2017.

[PV16] J. Pantone and V. Vatter. Growth rates of permutation classes: categorization up to the uncountability threshold. preprint, arXiv:1605.04289, 05 2016.

[Raz07] A. A. Razborov. Flag algebras. *The Journal of Symbolic Logic*, 72(04):1239–1282, 2007.

[Raz13] A. A. Razborov. Flag algebras: an interim report. In *The Mathematics of Paul Erdős II*, pages 207–232. Springer, 2013.

[Sli16] J. Sliačan. Permpack. http://jsliacan.github.io/permpack, 2016.

[Spe12] K. Sperfeld. *Semidefinite programming in extremal graph theory*. PhD thesis, Universität Rostock, 2012.

[Ste12] E. Steingrimsson. Some open problems on permutation patterns. *London Mathematical Society Lecture Note Series*, 409, 10 2012.

[Str93] W. Stromquist. Packing layered posets into posets. unpublished, 1993.

[V.15] Vatter V. Permutation classes. In Bóna M., editor, *Handbook of Enumerative Combinatorics*, pages 753–834. Chapman & Hall/CRC, 2015.

[Vat11] V. Vatter. Small permutation classes. *Proceedings of the London Mathematical Society (3)*, 103:879–921, 2011.

[Vat16] V. Vatter. Growth rates of permutation classes: from countable to uncountable. preprint, arXiv:1605.04297, 2016.

[Vau13] E. R. Vaughan. Flagmatic 2.0. http://jsliacan.github.io/flagmatic, 2013.

[Vol14] J. Volec. *Analytic methods in combinatorics*. PhD thesis, University of Warwick, Université Paris-Diderot, 2014.

[VW11] V. Vatter and S. Waton. On partial well-order for monotone grid classes of permutations. *Order*, 28:193–199, 2011.

[Wat07] S. Waton. *On permutation classes defined by token passing networks, gridding matrices and pictures: Three flavours of involvement*. PhD thesis, University of St Andrews, 2007.

[YFF+12] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata. *Latest Developments in the SDPA Family for Solving Large-Scale SDPs*, pages 687–713. Springer US, Boston, MA, 2012.