The enumeration of subclasses of the 321-avoiding permutations

Jinge Li

a thesis submitted for the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

4 January 2017

Abstract

This thesis is dedicated to the enumeration of subclasses of 321-avoiding permutations, using a combination of theoretical and experimental investigations.

The thesis is organised as follows:

Chapter 1 provides the necessary definitions and preliminaries, discusses the current state of research on the subject of enumerating Av(321) and its subclasses, then gives an introduction on the basic problem of containment check for 321-avoiding permutations, the process of which is used throughout our work.

The main results of this study are explained in Chapter 2 and 3. Chapter 2 focuses on the implementation aspects of enumerating 321-avoiding classes, where the main goal is to develop efficient algorithms to generate all permutations up to a certain length contained in classes of the form $\mathrm{Av}(321, \pi)$. The permutation counts are then used to guess the generating function by fitting a rational function to the computed data.

In Chapter 3, we deal with the more theoretical problem of enumerating 321-avoiding polynomial classes given a structural description. In particular, we propose a method which computes the grid class of such a class given its basis. We then use this information to enumerate the class using an improved version of a known algorithm.

Acknowledgements

I would like to express my sincere appreciation for all those who have helped me in various ways to make this thesis a reality.

First of all, I feel greatly indebted to my thesis supervisor, Michael Albert, for his great patience in seeing me through every stage of the present study. His unfailing support and encouragement, his expert advice and brilliant ideas, and his critical and timely comments have been an important force for me to get the work completed.

I would also like to thank the department of Computer Science at University of Otago for providing full funding for me to present my work at the 40th ACCMCC, which formed a part of this thesis.

I wish to thank Abbey College with its favourable and supportive environment that enabled me to pursue my postgraduate study with great peace of mind and deep concentration. A special thanks goes to the big external TV screen in the di Menna reading room, which I wrote most of this thesis on.

Last but not least, I wish to thank my parents for their financial support, understanding, love, care and all-around support when most needed. It is to them I dedicate this thesis.

Contents

1 Introduction **1**
- 1.1 Basic concepts . . . 1
  - 1.1.1 Permutations and classes . . . 1
  - 1.1.2 Wilf-Equivalence . . . 3
  - 1.1.3 Substitution decomposition of permutations . . . 4
- 1.2 321-avoiding permutation classes . . . 5
- 1.3 Pattern containment in 321-avoiding permutations . . . 7

2 Enumerating $\mathrm{Av}(321, \pi)$ by approximation **13**
- 2.1 DFS without array updates (simple DFS) . . . 14
- 2.2 DFS with array updates (array DFS) . . . 16
- 2.3 Runtime comparison . . . 21
- 2.4 Guessing generating functions . . . 24
- 2.5 Observations and concluding remarks . . . 29

3 Permutation classes of polynomial growth **33**
- 3.1 Introduction . . . 33
- 3.2 General structure of polynomial classes in terms of peg permutations . . . 34
- 3.3 The Homberger-Vatter Algorithm . . . 37
- 3.4 Enumerating polynomial classes of the form $\mathrm{Av}(321, \beta)$ . . . 43
  - 3.4.1 Structure of maximal plus irreducibles . . . 43
  - 3.4.2 The algorithm . . . 50
  - 3.4.3 Examples . . . 59
- 3.5 Concluding remarks . . . 65

References **67**

A Conjectured generating functions **69**
- A.1 Generating functions of $\mathrm{Av}(321, \pi)$ for $|\pi| = 5$ . . . 69
- A.2 Generating functions of $\mathrm{Av}(321, \pi)$ for $|\pi| = 6$ . . . 70

List of Tables

2.1 mean runtime (in seconds) of different search algorithms on rigid π of length 5, for $m = 17$ 22
2.2 mean runtime (in seconds) of different search algorithms on rigid π of length 6, for $m = 15$ 24
2.3 mean runtime (in seconds) of different search algorithms on non-rigid π of length 5, for $m = 17$ 25
2.4 All permutations of length 4, 5, and 6 that satisfy Theorem 2.2 31

List of Figures

1.1 2315467 ∈ Av(321) 6
1.2 The permutation $\tau = 3\,1\,2\,4\,5\,8\,9\,6\,10\,7 \in \mathrm{Av}(321)$. ■: upper, ●: lower, ▲: fluid. 9

2.1 For $\pi = 23154$, the slot markings on $\tau = 1352647$ (left), and $\tau' = \tau 4^{+\epsilon} = 13627485$ (right) 18

3.1 Extending ρ to ρ' in Lemma 3.10 by inserting x. Case 1 (left). Case 2 (centre). Case 3 (right). 45
3.2 Extending ρ to ρ' in Proposition 3.11. The shaded regions forbid elements of ρ due to 321 avoidance (gray), choice of d_2 and d_1 (lime), and choice of v (orange). 47
3.3 The permutations $\tilde{\rho} \in \tilde{G}$ for the class $\mathcal{C}_{3,0}$ 61

Chapter 1

Introduction

In this chapter, we give an accessible introduction to the background of our work. Here we will define basic concepts, discuss previous results on the enumeration of subclasses of 321-avoiding permutations along with our direction and goals, and finally take a look at the basic problem of pattern matching which we will need in the upcoming chapters.

1.1 Basic concepts

We start our introduction by defining basic concepts which will be used throughout the thesis. First we look at permutations and permutation classes.

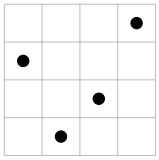
1.1.1 Permutations and classes

Definition 1.1. *For any integer $n \in \mathbb{P}$, a* permutation of length n *is a sequence $\pi = \pi_1\pi_2\dots\pi_n$ in which $\pi_i \in [n]$ and each integer of $[n]$ is used exactly once. We denote by $|\pi|$ the size of a permutation π, and $\mathcal{S}_n$ the set of all permutations of size n.*

Example 1.1. *The set $\mathcal{S}_3$ of all permutations of length 3 consists of the permutations 123, 132, 213, 231, 312, 321.*

Permutations can be represented by a set of points in the plane (up to arbitrary vertical and horizontal rescaling), where no two are on the same horizontal or vertical line. We define the *diagram* of a permutation $\pi \in \mathcal{S}_n$ as the set of points in the plane at

$(1, \pi_1), (2, \pi_2), \ldots, (n, \pi_n)$. The figure below shows the diagram of the permutation 3124.



The permutation 3124

Definition 1.2. *Two sequences $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ and $\beta_1\beta_2 \ldots \beta_n$ of distinct integers are* order isomorphic *(denoted $\alpha \sim \beta$) if*

$$\alpha_i < \alpha_j \text{ if and only if } \beta_i < \beta_j.$$

Definition 1.3. *Let $n, k \in \mathbb{P}$ with $k \leqslant n$. A permutation $\pi = \pi_1\pi_2 \ldots \pi_k$ is* contained as a pattern *in the permutation $\tau = \tau_1\tau_2 \ldots \tau_n$, denoted as $\pi \leqslant \tau$, if τ has a subsequence that is order isomorphic to π, that is, there exist some subsequence $1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$ such that*

$$\tau_{i_i}\tau_{i_2} \ldots \tau_{i_k} \sim \pi_1\pi_2 \ldots \pi_k.$$

Given permutations τ and π, we say that τ *avoids* π if π is not contained in τ. The set of all permutations which avoids π is denoted as $\mathrm{Av}(\pi)$.

Example 1.2. *The permutation $\tau = 3124$ contains $\pi = 213$ as the subsequence $\tau_{i_1}\tau_{i_2}\tau_{i_3} = 314$ is order isomorphic to $\pi = 213$. On the other hand, τ avoids the permutation $\sigma = 321$.*

We note that pattern containment is reflexive ($\pi \leqslant \pi$), transitive ($\sigma \leqslant \pi$ and $\pi \leqslant \tau$ implies $\sigma \leqslant \tau$) and anti-symmetric ($\pi \leqslant \tau$ and $\tau \leqslant \pi$ implies $\pi = \tau$). Thus the set of all permutations forms a *partially ordered set* (poset) with respect to the ordering defined above. A subset of this poset where no two elements are comparable using this ordering is called an *antichain*.

Definition 1.4. *A* permutation class *is a set of permutations closed downwards under pattern containment. In other words, a set $\mathcal{C}$ is a permutation class if and only if for any $\pi \in \mathcal{C}$, if $\sigma \leqslant \pi$, then $\sigma \in \mathcal{C}$.*

For any permutation class $\mathcal{C}$ there is a unique (and possibly infinite) antichain B such that $\mathcal{C}$ is the set of all permutations which avoid every permutation in B:

$$\mathcal{C} = \mathrm{Av}(B) = \{\pi : \beta \not\leqslant \pi \text{ for all } \beta \in B\}.$$

This antichain is called the *basis* of $\mathcal{C}$. If B is finite, we say $\mathcal{C}$ is *finitely based*.

Example 1.3. $\mathrm{Av}(321, 12354)$ *is a permutation class where all its permutations avoid both* 321 *and* 12354*. The basis of the class is* $\{321, 12354\}$.

Permutation classes are one of the core subjects in the study of permutation patterns. As we will see later in this thesis, our study focuses mainly on enumerating classes of the form $\mathrm{Av}(321, \pi)$ where π is some 321-avoiding permutation. Here *enumerating* a class $\mathcal{C}$ means finding the number $|\mathcal{C}_n|$ of elements of each length n in the class.

Definition 1.5. *The* generating function *of a permutation class* $\mathcal{C}$ *is defined by*

$$\sum_{n=0}^{\infty} |\mathcal{C}_n| \, x^n = \sum_{\pi \in \mathcal{C}} x^{|\pi|}$$

where $\mathcal{C}_n$ *denotes the set of permutations in* $\mathcal{C}$ *of length* n.

1.1.2 Wilf-Equivalence

The *growth rate* of a class $\mathcal{C}$, which indicates how fast $|\mathcal{C}_n|$ increase as n increases, is an important notion closely related to its enumeration. For classes where no exact enumeration is known, growth rate estimates can be very useful in approximating the number of permutations in them.

Definition 1.6. *Let* $\mathcal{C}$ *be a permutation class. The* (upper) growth rate *of* $\mathcal{C}$ *is defined as the limit*

$$\limsup_{n \to \infty} \sqrt[n]{|\mathcal{C}_n|}.$$

Definition 1.7. *Given two permutations* π *and* τ*, we say* π *and* τ *are* Wilf-equivalent *if* $|\mathrm{Av}_n(\pi)| = |\mathrm{Av}_n(\tau)|$ *for all* $n \in \mathbb{N}$*. The set of all Wilf-equivalent permutations form a* Wilf class.

It is in general hard to show Wilf-equivalence between a pair of permutations. However, there are trivial equivalences which arise from symmetries, such as the ones shown below:

Definition 1.8. *Let* $\pi = \pi_1 \pi_2 \dots \pi_n$ *be a permutation. The* reverse*, the* complement*,*

and the inverse *of* π*, are defined respectively as*

$$\begin{aligned}(\pi^r)_i &= \pi_{n-i+1},\\ (\pi^c)_i &= n - \pi_i + 1, \text{and}\\ (\pi^{-1})_{\pi_i} &= i.\end{aligned}$$

It is easy to show that $|\mathrm{Av}_n(\pi)| = |\mathrm{Av}_n(\pi^s)|$ for all $n \in \mathbb{N}$ where s denotes any combination of the operations in $\{r, c, -1\}$. These operations also preserve pattern containment, in a way that $\pi \leqslant \tau$ implies $\pi^s \leqslant \tau^s$. Thus, the class Av(321) and Av(123) have the same enumeration, and any known results on Av(123) and its subclasses applies to Av(321) by a simple symmetric argument. It is also easy to see that the classes Av(132), Av(213), Av(231) and Av(312) are all symmetric, which means there are only two essentially different classes of the form $\mathrm{Av}(\pi)$ for $|\pi| = 3$.

Example 1.4. *The classes* $\mathrm{Av}(321, 1235467)$ *and* $\mathrm{Av}(321, 1243567)$ *are Wilf-equivalent by symmetry, since* 1235467 *and* 1243567 *are reverse-complement of each other, and* 321 *is the reverse-complement of itself.*

1.1.3 Substitution decomposition of permutations

We now turn our attention to the substitution decomposition of permutations, which says that every permutation can be recursively decomposed into some simple permutation using substitutions. Such classification gives us the framework for representing an infinite number of certain permutations in terms of a 'structural representative', which we will use for the definitions of grid classes and peg permutations later in Chapter 3.

Definition 1.9. *Let* $n, k \in \mathbb{P}$*, and let* $\pi \in \mathcal{S}_n$ *and* $\sigma \in \mathcal{S}_k$*. The* direct sum *of* π *and* σ*, written* $\pi \oplus \sigma$*, is the permutation defined by*

$$(\pi \oplus \sigma)_i = \begin{cases} \pi_i & \text{if } i \leqslant n \\ \sigma_{i-n} + n & \text{if } i > n. \end{cases}$$

The skew sum $\pi \ominus \sigma$ *is similarly defined by*

$$(\pi \ominus \sigma)_i = \begin{cases} \pi_i + k & \text{if } i \leqslant n \\ \sigma_{i-n} & \text{if } i > n. \end{cases}$$

A sum-indecomposable *(resp.* skew-indecomposable*) permutation is one which cannot be written as a direct (resp. skew) sum.*

Example 1.5. *The permutation* $1\oplus 1\oplus 1\oplus 21\oplus 1\oplus 1$*, which we abbreviate by* $1^3\oplus 21\oplus 1^2$*, is the permutation* 1235467*.*

Definition 1.10. *Let* $\pi = \pi_1\pi_2\dots\pi_n$ *be a permutation. An* interval *of* π *is a sequence of contiguous indices* $\pi_i\pi_{i+1}\dots\pi_{i+k}$ *where its values are also contiguous. An interval is* monotone *if its entries are strictly increasing or decreasing. We say* π *is* plus *(resp.* minus*)* irreducible *if it contains no monotone increasing (resp. decreasing) interval, and* π *is* irreducible *if it is both plus and minus irreducible.*

Example 1.6. *The subsequence* 12 *in the permutation* 3124 *is a monotone interval which is increasing. Hence* 3124 *is not plus irreducible.*

Definition 1.11. *A permutation* $\pi \in \mathcal{S}_n$ *is said to be* simple *if the only intervals it contain are the trivial ones (i.e. have length* 1 *and* n*).*

Definition 1.12. *Given* $\pi = \pi_1\pi_2\dots\pi_n$ *and non-empty permutations* $\alpha_1,\dots,\alpha_n$*, the* inflation *of* π *by* $\alpha_1,\dots,\alpha_n$*, denoted* $\pi[\alpha_1,\dots,\alpha_n]$*, is the permutation obtained by replacing each entry* π_i *by an interval that is order isomorphic to* α_i*, while maintaining the relative order of the intervals themselves.*

Example 1.7. *For any two permutations* π *and* σ*,* $\pi\oplus\sigma = 12[\pi,\sigma]$ *and* $\pi\ominus\sigma = 21[\pi,\sigma]$*.*

Example 1.8. *For* $\pi = 3142$*,* $\alpha_1 = \alpha_3 = 1$*,* $\alpha_2 = 321$*,* $\alpha_4 = 12$*, we have* $3142[1, 321, 1, 12] = 6\,321\,7\,45$*.*

Theorem 1.1 (Substitution Decomposition [AA05])**.** *Every permutation* τ *can be written as the inflation of a unique simple permutation. Furthermore, if* $\tau = \pi[\alpha_1,\dots,\alpha_m]$*, where each* α_i *is a permutation of length* $\geqslant 1$ *and* $m \geqslant 4$*, then the permutations* α_i *are also uniquely determined.*

1.2 321-avoiding permutation classes

In this section we give an overview on the permutation classes central to our study - the class Av(321) and its subclasses.

For the class Av(321), we can describe its structure using the following proposition, which is part of the forklore of permutation patterns.

Proposition 1.2. *Every permutation of the class* Av(321) *can be partitioned into two increasing subsequences.*

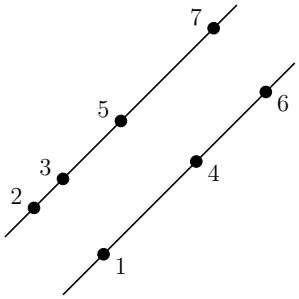


Figure 1.1: 2315467 $\in$ Av(321)

Proof. Label each entry of $\pi \in \mathrm{Av}(321)$ by the length of the longest decreasing sequence ending at that entry. Clearly only the labels 1 and 2 are used. By construction, the entries with the same label must form an increasing sequence. □

Watson proved in his thesis the following geometric version of Proposition 1.2:

Proposition 1.3 ([Wat07])**.** *Every permutation of the class* Av(321) *can be drawn on any two parallel lines of positive slope, as shown in Figure 1.1.*

Example 1.9. *The permutation* 2315467 *is contained in the class* Av(321)*. We can generate the permutation by adding points onto the two increasing lines as shown in Figure 1.1.*

In terms of enumeration, it is well known that $|\mathrm{Av}_n(321)| = c_n$, where c_n is the nth Catalan number. Interestingly this gives the same enumeration as the class Av(231).

However, unlike Av(231) whose subclasses are usually 'well behaved', the subclasses of Av(321) are in general not very tractable. There are uncountably many subclasses of Av(321), some of which are non-rational. Moreover, only very few subclasses of Av(321) have known exact enumerations, hence growth rate estimates are often relied upon in the study of such classes.

To simplify matters, in this thesis we will only look at classes of the form $\mathrm{Av}(321, \pi)$. Such classes has the merit of having rational generating functions, as have been proven by Theorem 1.1 of [ABRV16], which said that the generating functions of all finitely based subclasses of Av(321) are rational. This theorem potentially made it easier to approximate the number of permutations in a class $\mathcal{C} = \mathrm{Av}(321, \pi)$, as we may now be able to fit a rational function given a sufficient number of terms $|\mathcal{C}_n|$ using the equation

$$\frac{P(x)}{Q(x)} = \sum_{n=0}^{\infty} |\mathcal{C}_n|\, x^n.$$

Clearly, the more terms $|\mathcal{C}_n|$ we have, the more likely it is to give a good estimate on the rational generating function, which can then be used to generate an arbitrary number of (guessed) terms. As we currently only have complete enumerations of $\mathrm{Av}(321, \pi)$ for π of length up to 4, this 'guessing the rational function' approach may help us expand our knowledge on the generating functions or the growth rate estimates of classes where $|\pi| \geqslant 5$. We will investigate into this in Chapter 2, by first developing efficient algorithms which generate and count all permutations in $\mathcal{C}_n$ up to some sufficiently large n, then fit a rational function to the computed data using a linear algebraic approach.

In terms of structure, not much is known about subclasses of $\mathrm{Av}(321)$ in general. We do however have partial description on one particular type of subclasses - the polynomial classes with two basis elements. Albert, Atkinson and Brignall [AAB07] showed that these classes have the form $\mathrm{Av}(321, \beta)$, where $\beta = 1^k \oplus 21 \oplus 1^l$. In the same work they also provided degree bounds on irreducible elements of these classes. However, this still does not answer the important question of what such classes 'look like'. In fact, this question on the geometric structure of 321-avoiding polynomial classes has become even more important when Homberger and Vatter recently proposed in [HV15] an algorithm which enumerates any polynomial class given its grid class representation, as we can now enumerate these classes based on their geometric structure. We will solve this question in Chapter 3, by providing a method to compute the grid class of any 321-avoiding polynomial class given its basis. We then use an improved version of the Homberger–Vatter algorithm to enumerate such classes.

1.3 Pattern containment in 321-avoiding permutations

Before moving on to our main works, we would like to take a look at the classic problem of pattern matching (aka. pattern containment) which we will encounter time and again in the study of permutation patterns. The problem asks, given a *pattern* permutation π and a *text* permutation τ, whether π is contained in τ.

The pattern matching problem was first shown to be NP-complete on general input by Bose, Buss and Lubiw [BBL98]. Recently, Jelínek and Kynčyl [JK16] have showed much stronger results namely the problem is NP-complete for $\tau \in \mathrm{Av}(4321)$, $\pi \in \mathrm{Av}(321)$. For τ and π both in Av(321) however, it has been shown by Albert, Lackner, Lackner and Vatter [ALLV15] that there exist an algorithm which solves the pattern matching problem in $O(kn)$ time.

Theorem 1.4 ([ALLV15], Theorem 1)**.** *Given* 321*-avoiding permutations* τ *of size* n *and* π *of size* k*, there is an* $O(kn)$*-time algorithm which determines whether* τ *contains* π*.*

The $O(kn)$ bound presented in the above theorem is a good theoretical improvement over the naive bound $O(n^{k-1})$ on the runtime of the conventional containment check, which is based on exhaustive left to right scans for an occurrence of the pattern in the text. However, theoretical improvements do not always translate to runtime improvements in practice, thus in Chapter 2 we will conduct several tests to determine whether the algorithm from [ALLV15] is more efficient than the conventional check, for the purpose of enumerating the classes $\mathrm{Av}(321, \pi)$.

We now briefly go over the main steps of one part of the containment check algorithm given in [ALLV15] which we will use in Chapter 2. For a complete description we refer the reader to the paper [ALLV15] itself. Here we focus on some of the arrays constructed in the algorithm, in particular the *type array* and the *next arrays*, which will undergo repeated update in the *array DFS* implementation of the $\mathrm{Av}(321, \pi)$ enumeration algorithm described in Chapter 2.2.

First let us make a few definitions. From Section 1.2 we see that any permutation π in the class Av(321) can be partitioned into two increasing subsequences. One of them contains all elements that form a 2 in a copy of 21, which we call the *upper elements* of π and denote by U_π; and the other contains all elements that form a 1, which we call the *lower elements* of π and denote by L_π. Elements which are either upper or lower elements of π are called *rigid elements*, and we refer to elements that are non-rigid (i.e. those that do not participate in a copy of 21) as *fluid elements*. An example of the upper-lower-fluid element decomposition of a 321-avoiding permutation is given in Figure 1.2. A permutation π is a *rigid permutation* if all of its elements are rigid (i.e. if $\pi = U_\pi \cup L_\pi$). A map $f\colon \pi \to \tau$ is called a *rigid mapping* if f maps upper (resp. lower) elements of π to upper (resp. lower) elements of τ.

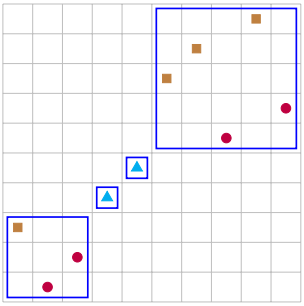


Figure 1.2: The permutation $\tau = 3\,1\,2\,4\,5\,8\,9\,6\,10\,7 \in \mathrm{Av}(321)$.
■: upper, ●: lower, ▲: fluid.

For our purpose, we will only consider the case where the text τ is any 321-avoiding permutation while the pattern π is rigid. In this case any embedding we found must be a rigid mapping. To handle fluid elements in the pattern requires some modification of the algorithm for the rigid case and is described in [ALLV15].

Given an element σ_i of some 321-avoiding permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_m$, we first define the *type* of σ_i, $T(\sigma_i)$ to be U if $\sigma_i \in U_\sigma$, L if $\sigma_i \in L_\sigma$ and F if σ_i is fluid. The *type array* of the permutation σ, denoted by T_σ, is then the array where each slot $T[i] = T(\sigma_i)$. For example, the type array of the permutation shown in Figure 1.2 is

$$T_{3\,1\,2\,4\,5\,8\,9\,6\,10\,7} = [U, L, L, F, F, U, U, L, U, L]. \tag{1.1}$$

Given two points, x and y, in U_σ (resp. L_σ), we say $x < y$ if y lies above and to the right of x. We define for any element $x \in \sigma$ the element immediately to its left, right, above and below by $x^{\blacktriangleleft}$, $x^{\blacktriangleright}$, $x^{\blacktriangle}$, and $x^{\blacktriangledown}$ respectively. We also define $x^{\blacktriangleright}_U$ to be the leftmost element of U_σ that is to the right of x, and similarly the corresponding notations $x^{\blacktriangleright}_L$, $x^{\blacktriangle}_U$ and $x^{\blacktriangle}_L$. In all cases, if no such element exists we represent it by $\perp$.

We then define the arrays $N^{\blacktriangleright}_U(\sigma)$, $N^{\blacktriangleright}_L(\sigma)$, $N^{\blacktriangle}_U(\sigma)$, and $N^{\blacktriangle}_L(\sigma)$ where each slot of the array is $N^a_b(\sigma)[i] = (\sigma_i)^a_b$ for $a \in \{\blacktriangleright, \blacktriangle\}$ and $b \in \{U, L\}$. We collectively call these arrays the *next arrays* of the permutation σ. Intuitively speaking, the next arrays of σ returns the next element to the right/above each of σ's entry that is an upper/lower element. For example, the next arrays of the permutation shown in Figure 1.2 are

$$N_U^{\blacktriangleright}(\tau) = [\tau_6, \tau_6, \tau_6, \tau_6, \tau_6, \tau_7, \tau_9, \tau_9, \perp, \perp]$$
$$N_L^{\blacktriangleright}(\tau) = [\tau_2, \tau_3, \tau_8, \tau_8, \tau_8, \tau_8, \tau_8, \tau_{10}, \tau_{10}, \perp]$$
$$N_U^{\blacktriangle}(\tau) = [\tau_6, \tau_1, \tau_1, \tau_6, \tau_6, \tau_7, \tau_9, \tau_6, \perp, \tau_6]$$
$$N_L^{\blacktriangle}(\tau) = [\tau_8, \tau_3, \tau_8, \tau_8, \tau_8, \perp, \perp, \tau_{10}, \perp, \perp].$$

Now suppose that f is any rigid mapping from π to τ. We say an element $x \in \pi$ *is a problem* if

$$f(x) < \max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\}. \tag{1.2}$$

Intuitively, $x \in \pi$ is a problem if the element $f(x)$ it mapped to in τ is too far left compared to $f(x^{\blacktriangleleft})$, or too low compared to $f(x^{\blacktriangledown})$. Let $P(f)$ be the set of problems for f. It is easy to see that f is an embedding if and only if $P(f)$ is empty.

We now describe the steps of the algorithm, along with the example where $\pi = 215364$, and $\tau = 3\,1\,2\,4\,5\,8\,9\,6\,10\,7$ which is the permutation shown in Figure 1.2. Given a pattern π and a text τ, we first precompute the type arrays of π and τ and the four next arrays of τ. Then we construct the initial mapping f_0 by mapping from left to right each of the i^{th} upper (resp. lower) element of π to the i^{th} upper (resp. lower) element of τ. If some upper/lower element of π does not map to any element of τ, then no rigid mapping exists from π to τ and the algorithm terminates.

For our example, the type array of π is

$$T_{215364} = [U, L, U, L, U, L].$$

The type array for τ was already given by Equation (1.1). The initial mapping f_0 is then

$$f_0(215364) = \tau_1\tau_2\tau_6\tau_3\tau_7\tau_8.$$

Next we construct as a queue the problems $P(f_0)$ of the initial mapping, by adding each $\pi_i \in \pi$ that satisfies Equation (1.2) to the queue. For instance, π_1 is not a problem

in our example, since

$$f(\pi_1^{\blacktriangleleft})^{\blacktriangleright}_{T(\pi_1)} = f(\bot)^{\blacktriangleright}_U = \bot;$$
$$f(\pi_1^{\blacktriangledown})^{\blacktriangle}_{T(\pi_1)} = f(\pi_2)^{\blacktriangle}_U = (\tau_2)^{\blacktriangle}_U = \tau_1;$$
$$\text{Hence } f(\pi_1) = \tau_1 \not\leq \max\{\bot, \tau_1\} = \tau_1.$$

On the other hand, π_4 is a problem since $f(\pi_4) = \tau_3$ while $f(\pi_4^{\blacktriangleleft})^{\blacktriangleright}_{T(\pi_4)} = f(\pi_3)^{\blacktriangleright}_L = (\tau_6)^{\blacktriangleright}_L = \tau_8$, $f(\pi_4^{\blacktriangledown})^{\blacktriangle}_{T(\pi_4)} = f(\pi_1)^{\blacktriangle}_L = (\tau_1)^{\blacktriangle}_L = \tau_8$, and $\tau_3 < \tau_8$. It turns out that π_4 is the only initial problem here, thus $P(f_0) = \{\pi_4\}$.

Now we enter a while loop which repeatedly updates the mapping f, starting from f_0, until $P(f)$ is empty, or until there is no valid update for some problem in $P(f)$. In the former case the resulting f is then an embedding of π into τ, and in the latter case no embedding exists and thus $\pi \notin \tau$. The detailed steps are displayed in Algorithm 1 below.

Algorithm 1: Find a (minimum) embedding of π into τ, or demonstrate that no embedding exist.

Initialise: $f \leftarrow f_0$;

while *f is defined everywhere, and $P(f)$ is nonempty* **do**

Choose $x \in P(f)$;

Update: $f(x) \leftarrow \max\{f(x^{\blacktriangleleft})^{\blacktriangleright}_{T(x)}, f(x^{\blacktriangledown})^{\blacktriangle}_{T(x)}\}$;

Recompute: $P(f)$.

return f

In our running example, since $P(f_0) = \{\pi_4\}$, we pop π_4 from the queue, and update $f(\pi_4)$ to $\max\{f(\pi_4^{\blacktriangleleft})^{\blacktriangleright}_{T(\pi_4)}, f(\pi_4^{\blacktriangledown})^{\blacktriangle}_{T(\pi_4)}\} = \tau_8$. Recomputing $P(f)$ we get $P(f) = \{\pi_5, \pi_6\}$ with two new problems π_5 and π_6. The subsequent iterations of the while loop goes as follow:

choose π_5,

$f(\pi_5) \leftarrow \max\{f(\pi_5^{\blacktriangleleft})^{\blacktriangleright}_U, f(\pi_5^{\blacktriangledown})^{\blacktriangle}_U\} = \tau_9$,

$P(f) = \{\pi_6\}$;

choose π_6,

$f(\pi_6) \leftarrow \max\{f(\pi_6^{\blacktriangleleft})^{\blacktriangleright}_L, f(\pi_6^{\blacktriangledown})^{\blacktriangle}_L\} = \tau_{10}$,

$P(f) = \emptyset$.

The final embedding $f\colon \pi \to \tau$ is therefore

$$f(215364) = \tau_1\tau_2\tau_6\tau_8\tau_9\tau_{10} = 3\,1\,8\,6\,10\,7.$$

Chapter 2

Enumerating $\mathrm{Av}(321, \pi)$ by approximation

In this chapter we examine the enumeration of permutation classes of the form $\mathcal{C} = \mathrm{Av}(321, \pi)$ by approximation. In particular, we provide efficient search algorithms which compute the number of permutations of each length that avoid both 321 and π. We then present a method that uses the obtained numbers to guess the generating function of $\mathcal{C}$.

In Sections 2.1 and 2.2 we present two depth first search (DFS) algorithms which generate the permutations in the class $\mathcal{C}$. The first one, which we refer to as *simple DFS*, uses a direct recursive implementation without array updates, and is best used with the generic containment check. The second one, which we call *array DFS*, uses stack implementation with array updates, and is a theoretical improvement on the simple DFS when being used with the new containment check described in Algorithm 1.

Our goal is to compare the runtime of the two implementations, and find out which is more efficient in generating all permutations in the class $\mathcal{C}$ up to a predefined length. The motivation of our work comes from Algorithm 1 proposed in [ALLV15], which checks whether a pattern π of length k is contained in a text τ of length n in $O(kn)$ time, when both π and τ avoid 321. As the generic pattern checking algorithm has a much higher naive bound of $O(n^{k-1})$, it is natural to ask whether in practice we can generate permutations in $\mathrm{Av}(321, \pi)$ faster by using Algorithm 1 for containment check. The results of our comparison are presented in Section 2.3.

We finally guess the generating function of $\mathcal{C}$ in Section 2.4 from the data computed in the earlier sections using a linear algebraic approach. With this method, conjectured generating functions are found for most classes $\mathrm{Av}(321, \pi)$ where $|\pi| = 5, 6$, which we present in Appendix A.

2.1 DFS without array updates (simple DFS)

Let $\mathcal{C} = \mathrm{Av}(321, \pi)$ be a 321-avoiding permutation class, where π is a pattern of length k. Our goal in this chapter is to enumerate classes of the form $\mathcal{C} = \mathrm{Av}(321, \pi)$ with good approximations. This involves finding the number $|\mathcal{C}_m|$ of permutations contained in the class $\mathcal{C}$, up to a finite length limit $m \in \mathbb{N}$ that we deemed sufficiently large.

We thus want to compute for each $i \leqslant m$, the number of text τ of length i that avoid both 321 and π. For $i \leqslant k$ this can be done trivially: $|\mathcal{C}_i|$ is simply the number of 321-avoiding permutations of length i when $i < k$, which equals $|\mathrm{Av}(321)| - 1$ when $i = k$ since the only π containing permutation of length k is π itself.

To find the number $|\mathcal{C}_i|$ for length $i > k$ is more difficult. Usually this is accomplished by generating all texts τ avoiding both 321 and π. A straightforward way to generate all such texts is to use a breath first search (BFS), by iterating over every 321-avoiding τ of each length greater than k, and check whether $\pi < \tau$. However, this requires us to do containment check for every text permutation τ of length greater than k, which is inefficient since containment checks dominate the runtime of the algorithm.

Instead, we can prune the algorithm by eliminating unnecessary containment checks, by using depth first search (DFS) based on the following observation:

Observation 2.1. *In* $\mathrm{Av}(321)$*, every permutation* τ' *of length* $n + 1$ *can be obtained from some* τ *of length* n *by adding one element at the end.*

For convenience, we refer to the process of adding one element at the end described in Observation 2.1 simply as 'extension'. And for two text permutations τ and τ', we say $\tau' \succ \tau$ if τ' results from τ by one or more extension steps.

Following Observation 2.1, we can represent the elements of $\mathrm{Av}(321, \pi)$ up to length m as a tree of depth m, where the root node is the empty permutation, and each node of depth $i \leqslant m$ represents a permutation in $\mathrm{Av}(321, \pi)$ of length i. For each node in the tree representing a permutation τ, we represent the one point extensions of τ in

$\mathrm{Av}(321, \pi)$ by its children. Here we can safely ignore any extension τ' of τ not in the class $\mathrm{Av}(321, \pi)$ since, for any permutation τ not in the class $\mathrm{Av}(321, \pi)$, no extension of τ can be in $\mathrm{Av}(321, \pi)$. Hence computing all nodes in the tree up to depth k gives the collection of all permutations in $\mathrm{Av}(321, \pi)$ up to length k.

As noted earlier, the computation of nodes of depth k or smaller is straightforward. We thus begin our traversal on the depth k nodes, and denote by $\mathcal{T}$ the set of permutations represented by them. i.e. $\mathcal{T} = \{\tau : \tau \in \mathrm{Av}(321, \pi), |\tau| = k\}$. Let $\tau = \tau_1\tau_2 \dots \tau_n$ be a text permutation of length $n \geqslant k$, and j be an integer with $0 \leqslant j \leqslant n$. We denote by $j^{+\epsilon}$ an element valued just above j, and $\tau'_j = \tau j^{+\epsilon}$ the extension of τ by adding $j^{+\epsilon}$ at the end. Here τ can be extended at the end by $n + 1$ different values of $j^{+\epsilon}$, each of them lies in a region which we call the j^{th} *slot* of τ. In other words, the j^{th} slot of τ for $0 < j < n$ is the region between the j^{th} and the $(j + 1)^{\text{st}}$ element by value, while the 0^{th} and n^{th} slots are respectively the regions below the least element and above the greatest element.

We now describe how the DFS works. Starting from some node in $\mathcal{T}$, when a node τ of length n is visited, we compute its children by including all extensions of τ that are in $\mathrm{Av}(321, \pi)$. Here 321-avoidance is assured by extending only on each slot above the last lower element $\tau(\ell_{\text{last}})$ of τ. That is, we create the extension $\tau'_j = \tau j^{+\epsilon}$ for each j with $\tau(\ell_{\text{last}}) \leqslant j \leqslant n$. We then ensure π-avoidance by checking if each extension τ'_j contains π, and only keeps the ones that do not. Note in addition that when π is rigid, then $\tau'_n \not> \pi$ whenever $\tau \not> \pi$. Thus we can get a small runtime reduction on rigid π by excluding the containment checks for $j = n$. After all children of τ are computed, we traverse to a child node of τ in a depth first manner, and repeat the above operations.

The search terminates when every node of depth $\leqslant m$ has been visited. The number of nodes of each depth i then gives the coefficient $|\mathcal{C}_i|$ in the generating function of $\mathcal{C}$. To count the nodes, we use an array A of length m, where each $A[i]$ is the number of permutations of length i that avoid 321 and π. For $i \leqslant k$ the count $A[i]$ can be computed directly. For $i > k$, whenever we visit a node of depth i in the DFS, we increment $A[i]$ by 1. We thus obtain a complete array when the search terminates, and we have $|\mathcal{C}_i| = A[i]$ for each $i \leqslant m$.

A detailed illustration of the algorithm is shown in Algorithm 2 below.

Algorithm 2: Compute the number of texts τ of each length i avoiding π using simple DFS, for $k < i \leqslant m$

Input : A pattern π of length k, and an upper limit m on the length of τ

Output: An array A of length m, where the i^{th} entry denotes the number of length i texts containing π

Initialise: $\mathcal{T} := \{\tau : \tau \in \mathrm{Av}(321, \pi), |\tau| = k\}$

foreach $i < k$ **do**
  $A[i] = |\mathrm{Av}(321)| \cap \mathcal{S}_i$

$A[k] = |\mathrm{Av}(321) \cap \mathcal{S}_k| - 1$

foreach $\tau \in \mathcal{T}$ **do**
  `generating321Avoidance(`τ`)`:
  if *length*$(\tau) < m$ **then**
    for $j = \tau(\ell_{last})$ **to** *length*(τ) **do**
      $\tau' := \tau j^{+\epsilon}$;
      if $\pi \not\leq \tau'$ **then**
        $A[\mathrm{length}(\tau')]$++;
        `generating321Avoidance(`τ'`)`

return A

2.2 DFS with array updates (array DFS)

Another advantage of using DFS by extending the permutations $\tau \in \mathcal{T}$ is that, since the containment check described in Algorithm 1 uses arrays that store the element type and the values x_b^a, $a \in \{\blacktriangleright, \blacktriangle\}$, $b \in \{U, L\}$ for each element $x \in \tau$, we can embed the new containment checks algorithm within the search, so that whenever we extend τ to a π-avoiding τ'_j, those arrays of τ' can be obtained from simply updating the respective arrays of τ. This means we do not need to recompute the arrays for each containment check and thus reduces the runtime for the algorithm.

Furthermore, the array update approach allow us to eliminate more containment checks by passing information from τ to τ'_j about when not to extend: we do so by associating every permutation τ with a boolean array of slot markings, which we denote B_τ. For each of the j^{th} slot of τ, we assign $B_\tau[j] = \mathsf{false}$ if extending τ on this slot produces a permutation τ' that contains either 321 or π, and $B_\tau[j] = \mathsf{true}$ if τ' avoids both, or if

further checks are needed to determine whether $\pi < \tau'$.

An example of this is shown in Figure 2.1, where $\pi = 23154$ and $\tau = 1352647$. We can extend τ since it does not contain π. Now the last lower element $\tau(\ell_{\text{last}})$ is 4, thus extending τ by any element smaller than 4 is not allowed as it will create an instance of 321. Extending τ on each slot greater than $\tau(\ell_{\text{last}})$ gives:

$$\begin{aligned}
\tau'_4 &= 1352647\,4^{+\epsilon} = 13627485 \not> 23154 \\
\tau'_5 &= 1352647\,5^{+\epsilon} = 13527486 > 23154 \\
\tau'_6 &= 1352647\,6^{+\epsilon} = 13526487 > 23154 \\
\tau'_7 &= 1352647\,7^{+\epsilon} = 13526478 \not> 23154
\end{aligned}$$

Here τ'_4 and τ'_7 are the allowed extensions of τ, the slots of which are marked true and are represented by the symbol ✓ in the first diagram of Figure 2.1. The rest of the slots are marked false and are represented by ✗. The corresponding boolean array of slot markings for τ is then

$$B_\tau = [\mathsf{false}, \mathsf{false}, \mathsf{false}, \mathsf{false}, \mathsf{true}, \mathsf{false}, \mathsf{false}, \mathsf{true}]$$

As noted earlier, we only need to extend on slots marked true. Furthermore, we can store the relative positions of the false markings for τ, so that as we extend τ to larger permutations, the slots marked with false will carry over in the way described in the following observation:

Observation 2.2. *Let $\tau = \tau_1\tau_2\ldots\tau_n$ be a text permutation of length n. If for any indices a, b with $1 \leqslant a, b \leqslant n$ that the slots between τ_a and τ_b are marked as* false*, then for every extension $\tau' \succ \tau$, the slots between τ'_a and τ'_b will be marked as* false*.*

Intuitively speaking, Observation 2.2 says that however we extend τ at the end, the slots marked as false will have the same relative positions as the original elements of τ.

Thus we can update the false slots in a similar way as adding an element. When an element $j^{+\epsilon}$ is extended to τ, the slot which contains $j^{+\epsilon}$ splits into two in τ'_j - one lies immediately above $j^{+\epsilon}$ and one immediately below.

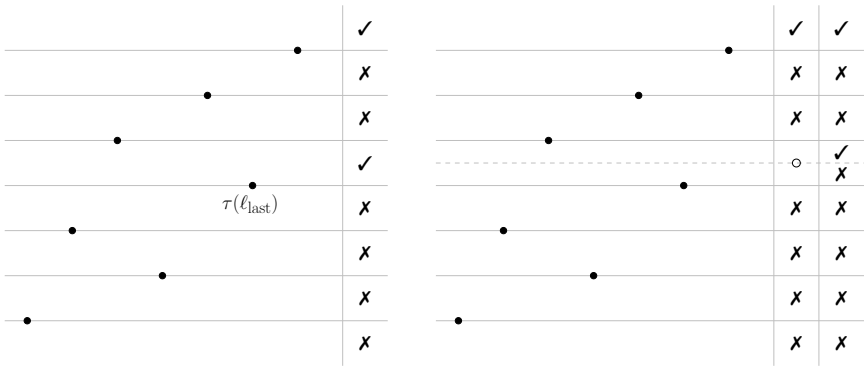


Figure 2.1: For $\pi = 23154$, the slot markings on $\tau = 1352647$ (left), and $\tau' = \tau 4^{+\epsilon} = 13627485$ (right)

If $j^{+\epsilon}$ is not the maximal element in τ'_j, it must be a lower element. Thus the newly created slot immediately below $j^{+\epsilon}$, along with every other slots lying below it, must be marked false. Now due to the extra slot created in τ'_j above $j^{+\epsilon}$, every slot in τ that lies above $j^{+\epsilon}$ will have its position move up by one in τ'_j. Hence for any non-maximal $j^{+\epsilon}$, we have

$$B_{\tau'_j}[i] = \begin{cases} \mathsf{false} & \text{for } i \leqslant j \\ B_\tau[i-1] & \text{otherwise} \end{cases} \tag{2.1}$$

If $j^{+\epsilon}$ is the maximal element in τ'_j, then τ'_j is a fluid element and the last lower element $\tau'_j(\ell_{\text{last}})$ remain the same as $\tau(\ell_{\text{last}})$. It is then clear that every slot in τ will carry over to τ'_j in the exact same position.

In our running example, after extending τ to $\tau'_4 = 13627485$, the slot in τ containing $4^{+\epsilon}$ splits into two in τ'_4, separated by the newly added element 5 of τ'_4. Note that 5 is also the last lower element of τ'_4, thus the 0^{th} to 4^{th} slots below 5 must be marked false.

In addition, since the 5^{th} and 6^{th} slot in τ are marked false, we do not need to extend on those positions in τ or on any of its extensions. In τ'_4 they correspond to the 6^{th} and the 7^{th} slots respectively, hence we can also mark them as false.

Extending τ'_4 on the rest of the slots in $\tau'_4(\ell_{\text{last}})$ gives:

$$\tau''_{4,5} = 13627485\,5^{+\epsilon} = 137284956 \not\succ 23154$$
$$\tau''_{4,8} = 13627485\,8^{+\epsilon} = 136274859 \not\succ 23154$$

thus we mark both the 5^{th} and 8^{th} slots as true, as seen from the second diagram of Figure 2.1. The corresponding boolean array for τ'_4 is

$$B_{\tau'_4} = [\mathsf{false}, \mathsf{false}, \mathsf{false}, \mathsf{false}, \mathsf{false}, \mathsf{true}, \mathsf{false}, \mathsf{false}, \mathsf{true}]$$

We now give a complete description of the DFS algorithm involving array updates.

As in the previous algorithm, we represent the elements of $\mathrm{Av}(321, \pi)$ as a depth m tree, where each node of depth $i \leqslant m$ represents a permutation of the same length. Here every node is also associated with the set of arrays (type, next, slot) of the permutation it represents, where the slot array provides a list of all possible children.

We initialise the set $\mathcal{T}$ of starting permutations in $\mathrm{Av}(321, \pi)$ of length k as in the previous algorithm, along with the type array, next arrays, and the slot array B_τ associated with each $\tau \in \mathcal{T}$. Here the slot arrays are computed by extending τ on each slot above $\tau(\ell_{\text{last}})$ and check for π containment, and assign $B_\tau[j] = \mathsf{true}$ if $\tau'_j \not\succ \pi$ and $B_\tau[j] = \mathsf{false}$ otherwise.

When a node τ is visited, we compute its children with the following steps: we first create the one point extension $\tau'_j = \tau j^{+\epsilon}$ on each of the j^{th} slot of τ where $B_\tau[j] = \mathsf{true}$. This ensures 321-avoidance for τ'_j as we discussed before, and excludes those extensions that are known to contain π from the slot array information of earlier nodes. We then perform π-containment check on each extension τ'_j, and if $\tau'_j \not\succ \pi$, include τ'_j as a child of τ, otherwise change $B_\tau[j]$ to false.

We now have a finalized slot array B_τ and a complete set of children for τ. Next we update the arrays for each child τ'_j from the arrays of τ. In particular, the slot array $B_{\tau'_j}$ is updated from B_τ according to Equation 2.1. When the update step is completed, we traverse to a child node of τ in a depth first manner, and the same operations repeat.

The search terminates when every node of depth $\leqslant m$ has been visited. As in the earlier algorithm, we keep track of the number of nodes of each depth using an array

A, which increments $A[i]$ by 1 upon visiting a node of depth i. The coefficient of the generating function of $\mathcal{C}$ is then given by $|\mathcal{C}_i| = A[i]$ for each $i \leqslant m$.

Algorithm 3: Compute the number of texts τ of each length i avoiding π using array DFS, for $k < i \leqslant m$

Input : A pattern π of length k, and an upper limit m on the length of τ

Output: An array A of length m, where the i^{th} entry denotes the number of length i texts containing π

Initialise:

$\mathcal{T} := \{\tau : \tau \in \mathrm{Av}(321, \pi), |\tau| = k\}$;

Compute type array, next arrays, and B_τ for each $\tau \in \mathcal{T}$;

Add every $\tau \in \mathcal{T}$ to the stack S

foreach $i < k$ **do**
  $A[i] = |\mathrm{Av}(321)| \cap \mathcal{S}_i$

$A[k] = |\mathrm{Av}(321) \cap \mathcal{S}_k| - 1$

`computeChildren(`τ`)`:

for $j = \tau(\ell_{last})$ **to** *length*(τ) **do**
  if $B_\tau[j] =$ true **then**
    $\tau'_j := \tau j^{+\epsilon}$;
    if $\pi < \tau'_j$ **then**
      $B_\tau[j] =$ false
    else
      add τ'_j to τ's set of children

Stack implementation of DFS:

while S *is not empty* **do**
  $v = S$.pop ;
  if *length*$(v) < m$ **then**
    foreach *children* v' *of* v *computed by* `computeChildren(`v`)` **do**
      S.push(v')
  A[length(v)]++

return A

For completeness, we also describe a small theoretical improvement for the containment check presented in Algorithm 1, which can be used when it is embedded in our DFS with array updates. We did not include this into Algorithm 3 proper however, since according to our runtime test, this theoretical improvement seems to have negli-

gible effect on the runtime of our DFS as a whole. Nevertheless, we feel this may be worth mentioning as it may become useful again in case of a future implementation improvement.

We know in our DFS every permutation τ' not in the initial set $\mathcal{T}$ is obtained by extending some τ at the end by one element. Thus in the π-involvement check for τ' we must make use of the last element of τ'. Then the construction of the initial mapping f_0 in Algorithm 1 can be done backwards, by assigning the last element of τ' to the last element of π of the same type, then mapping the rest of the elements from right to left. This saves one mapping computation in each f_0 construction.

As it turns out, if τ'' is an extension of τ' in $\mathrm{Av}(321, \pi)$, we can go one step further and claim that when checking whether $\pi < \tau''$, both of the last two elements of τ'' must be used. The reason is that since the extension $\tau'_j = \tau j^{+\epsilon}$ creates an extra slot in τ'_j just above $j^{+\epsilon}$, for any slot above $j^{+\epsilon}$ in τ'_j, the l^{th} slot of τ'_j corresponds to the $(l-1)^{\text{th}}$ slot of τ. Thus if $\tau''_{j,l} = \tau'_j l^{+\epsilon}$ only uses the last element $l^{+\epsilon}$ in the π-involvement check for $\tau''_{j,l}$, we can use the last element $(l-1)^{+\epsilon}$ in τ'_j instead, and get the same mapping for the π-involvement check for τ'_j. But then this was already checked when we visit the node τ'_j.

It is easy to see from the argument above that the using of last two elements only applies for nodes τ of length at least $k+2$, where k is the length of π.

2.3 Runtime comparison

We now come to the important question which we have been aiming for: is the new containment check given by Algorithm 1 an improvement from the old check, for the purpose of enumerating the class $\mathrm{Av}(321, \pi)$? Having described in the last subsections two different search algorithms - the simple DFS and the array DFS, we seek to answer this question by comparing the enumeration speed of each search–containment check combination, and determine if we can obtain an overall better runtime when the new containment check is used.

We thus have five subjects in our test: BFS, old containment check with simple and array DFS, and new containment check with these two DFS. All five are listed in Table 2.1, where we abbreviate the old and new containment checks by 'old' and 'new' respectively, and refer to the two DFS as 'simple' and 'array'. Note again that the new

π	BFS	old, simple	old, array	new, simple	new, array
21453	220	30.4	28.0	44.3	32.9
24153	626	132	133	180	141
31452	106	163	155	235	173
23451	128	191	182	244	202
41523	787	236	218	280	224
51234	609	285	390	266	213

Table 2.1: mean runtime (in seconds) of different search algorithms on rigid π of length 5, for $m = 17$

containment check given by Algorithm 1 only applies when the pattern π is rigid.

Theoretically speaking, we expect the runtime of each combination above to be based on two main factors. The first factor is the number of containment checks performed, since these checks dominate the runtime of the enumeration process. And the second is the runtime of each containment check, which depends on whether the old or the new algorithm is used. For the former, it is clear that array DFS requires the least number of checks, followed by simple DFS, while the BFS is least efficient in this category. For the latter, we note that in the case of simple DFS, the exact same search algorithm is used in both 'old, simple' and 'new, simple', and their only difference lies in the containment check embedded within the search. Thus the runtime difference between these two combinations should reflect the runtime difference between the old and new containment check algorithms.

As far as implementation is concerned, the coding and testing are done in Java, where we implemented the new containment check algorithm along with both simple and array DFS, as an extension to the program PermLab [Alb]. Here the enumeration algorithm takes a pattern π and an upper length limit m of $\tau \in \mathrm{Av}(321, \pi)$ as input, and outputs the array A together with the time taken to generate A. The testing of the code is done on an Intel Core i5-6300U processor machine.

In the first test, we choose rigid π of length 5, and enumerate the class $\mathcal{C} = \mathrm{Av}(321, \pi)$ up to elements of length $m = 17$ using the five aforementioned algorithm combinations. For each chosen π, we run each algorithm five times and report the average time in seconds with three significant figures. The results are shown in Table 2.1. Note that for every recorded mean in Table 2.1, the time taken for each of the five runs is within

10% from the listed number.

From the data we have a few minor observations:

- The BFS runs slower than any DFS implementation by a large margin, showing the exclusion of unnecessary containment checks in our DFS algorithms is effective.
- When being used with the new containment check, the array DFS has shown noticeable speed improvement over the simple DFS in every test. This is an expected outcome by design: as slot array updates reduced the number of containment checks, and updates on type and next arrays reduced array computation cost in the new containment check.
- On the old containment test however, array DFS has not shown significant improvement over simple DFS. Our speculation for this is that, since the old containment test only benefits from the slot array updates in array DFS, the object creations in the stack implementation in Java, which include creation of the slot array and the set of children at each node, offset some of the speed gains (or even outweigh that on rare occasions) obtained from reducing containment checks.

But perhaps the most important finding here is that, contrary to theoretical expectation, enumerating $\mathrm{Av}(321, \pi)$ using the new containment check is in most cases slower than using the old check, regardless of which DFS we use.

We first note that 'old, simple' is noticeably faster than 'new, simple', and as we discussed earlier, this implies that the old containment check by itself is faster than the new containment check on average. One possible reason for this is that the new check involves more array creations than the old check in its implementation, which can have an effect in a language like Java which is known for having slow object creations. And even though the naive bound on the runtime of a single instance of the old containment check is $O(n^{k-1})$, it may not have a significant impact for patterns and texts of short length (here we have $k = 5$ and $n \leqslant m = 17$).

Unfortunately, even the improvements by array DFS are not quite enough to overcome the speed deficiency from the new containment check, as we can see from the table, 'new, array' is still slightly slower than 'old, simple'.

To further confirm our finding, we give a second test on several rigid π of length 6, and enumerate the class $\mathrm{Av}(321, \pi)$ up to length 15, this time only with 'old, simple' and

	old, simple	new, array
$\pi = 231564$	34.5	34.8
$\pi = 214563$	42.5	47.4
$\pi = 214365$	55.2	58.9
$\pi = 235164$	60.9	82.0
$\pi = 512364$	134	102
$\pi = 235614$	74.3	141
$\pi = 456123$	82.3	150

Table 2.2: mean runtime (in seconds) of different search algorithms on rigid π of length 6, for $m = 15$

'new, array'. The average runtime over five runs is shown in Table 2.2, which again reports shorter runtime for the old check.

The above result for rigid π, coupled with the fact that the new containment check for non-rigid π requires additional treatments on top of Algorithm 1 to make it work (we refer the reader to Section 3 of [ALLV15] for a detailed description of the non-rigid case), lead us to conclude that it is unlikely to be worthwhile to improve the implementations using new containment check. This is because the degree of improvement which could be achieved over the simple brute force ones is likely to be relatively small, and in the presence of a combinatorial explosion such small margins are not worth pursuing.

Finally we perform a similar runtime test for non-rigid π, on BFS and the old containment check with simple DFS. The results are shown in Table 2.3.

Once again the DFS algorithm produces a much shorter runtime than the BFS. It is also worth noting, that compared to rigid π of the same length, for non-rigid π it is often much faster to enumerate $\mathrm{Av}(321, \pi)$ up to the same length limit m, due to the fact that there are generally fewer elements in $\mathrm{Av}(321, \pi)$ of each length when π is non-rigid.

2.4 Guessing generating functions

The DFS algorithms described in the previous sections enable us to compute the first few terms of the generating function of $\mathcal{C} = \mathrm{Av}(321, \pi)$. However, this is nowhere close

	BFS	old, simple
$\pi = 12354$	82.4	2.20
$\pi = 12435$	85.0	4.65
$\pi = 13425$	154	9.68
$\pi = 12453$	161	8.51
$\pi = 21354$	123	23.0
$\pi = 13254$	152	29.8
$\pi = 15234$	303	75.1
$\pi = 31425$	334	82.8

Table 2.3: mean runtime (in seconds) of different search algorithms on non-rigid π of length 5, for $m = 17$

to finding the entire generating function which normally has arbitrary large terms. Fortunately, as pointed out in the next theorem by Albert, Brignall, Ruškuc and Vatter [ABRV16], a class of the form $\mathrm{Av}(321, \pi)$ has a nice enough structure that it has a rational generating function, i.e. its GF can be represented by the quotient $P(t)/Q(t)$ of two polynomial functions P and Q.

Theorem 2.1 (Theorem 1.1 in [ABRV16])**.** *If a proper subclass of the* 321*-avoiding permutations is finitely based then it has a rational generating function.*

Theorem 2.1 means that in principle once we have enough data for a class $\mathrm{Av}(321, \pi)$, we will be able to fit a rational function to it. We thereby illustrate a method which guesses the polynomials $P(t)$ and $Q(t)$ by first guessing their degrees, then computing their coefficients from the terms obtained from our DFS. Note that the method we use is by no means the only way to guess the polynomials, another (and perhaps much slicker) way of doing it is to use Padé approximants.

Recall that the generating function of $\mathcal{C}$ is given by

$$\sum_{n=0}^{\infty} |\mathcal{C}_n|\, t^n = 1 + c_1 t + c_2 t^2 + \cdots .$$

The rational generating function of a class $\mathrm{Av}(321, \pi)$ can then be written as

$$\frac{P(t)}{Q(t)} = 1 + c_1 t + c_2 t^2 + c_3 t^3 + \cdots$$
$$P(t) = Q(t)(1 + c_1 t + c_2 t^2 + c_3 t^3 + \cdots)$$

Let's say if we guess both $P(t)$ and $Q(t)$ have degree 2, then $P(t) = p_0 + p_1 t + p_2 t^2$ and $Q(t) = q_0 + q_1 t + q_2 t^2$ and we have

$$p_0 + p_1 t + p_2 t^2 = (1 + c_1 t + c_2 t^2 + c_3 t^3 + \cdots)(q_0 + q_1 t + q_2 t^2)$$

Since the coefficients of t^3 and all higher powers of t on the left hand side are 0, and since we assume that some initial sequence of c_i are known, we obtain a sequence of equations for the three variables q_0, q_1 and q_2

$$0 \cdot t^3 = (c_3 q_0 + c_2 q_1 + c_1 q_2) t^3$$
$$0 \cdot t^4 = (c_4 q_0 + c_3 q_1 + c_2 q_2) t^4$$
$$0 \cdot t^5 = (c_5 q_0 + c_4 q_1 + c_3 q_2) t^5$$

Generalising to the case where we guess $P(t)$ and $Q(t)$ have degree m, we get the matrix equation

$$\begin{bmatrix} c_{m+1} & c_m & \cdots & c_0 \\ c_{m+2} & c_{m+1} & \cdots & c_1 \\ \vdots & \vdots & \ddots & \vdots \\ c_{2m+1} & c_{2m} & \cdots & c_{m+1} \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.2}$$

where each c_i is a known term obtained from the DFS. We can now solve the equation to find the vectors for $Q(t)$. Note that we will only obtain unique solution for the correct guess of degree (hence correct vector size) of $Q(t)$ - no non-trivial solution can exist if the degree we guessed is smaller than the actual degree of $P(t)$ and $Q(t)$, and infinitely many solutions exist if our guessed degree is larger. Thus in practice we may need to do some trial and error with our guesses, which involve increasing or decreasing the degree guess according to the number of solutions obtained from solving Equation (2.2).

We can then find $P(t)$ from the vectors of $Q(t)$ using the following:

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_m \end{bmatrix} = \begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{m+1} & c_m & \cdots & c_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_m \end{bmatrix} \tag{2.3}$$

We shall illustrate the above steps with an example. Let $\mathcal{C} = \mathrm{Av}(321, 132564)$, the first 22 terms of its generating function are

$$131, 414, 1298, 3980, 11848, 34170, 95567, 259838, 688841, 1785854,$$
$$4540152, 11345952, 27929582, 67844842, 162876845, 386950370, 910721351.$$

We use the first 21 terms to get a 11×11 matrix for Equation (2.2) shown below, which is equivalent to guessing $P(t)$ and $Q(t)$ to have degree 10. Note that the number of terms the matrix uses is always an odd number.

$$\begin{bmatrix} 34170 & 11848 & 3980 & 1298 & 414 & 131 & 42 & 14 & 5 & 2 & 1 \\ 95567 & 34170 & 11848 & 3980 & 1298 & 414 & 131 & 42 & 14 & 5 & 2 \\ 259838 & 95567 & 34170 & 11848 & 3980 & 1298 & 414 & 131 & 42 & 14 & 5 \\ 688841 & 259838 & 95567 & 34170 & 11848 & 3980 & 1298 & 414 & 131 & 42 & 14 \\ 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 & 3980 & 1298 & 414 & 131 & 42 \\ 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 & 3980 & 1298 & 414 & 131 \\ 11345952 & 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 & 3980 & 1298 & 414 \\ 27929582 & 11345952 & 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 & 3980 & 1298 \\ 67844842 & 27929582 & 11345952 & 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 & 3980 \\ 162876845 & 67844842 & 27929582 & 11345952 & 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 & 11848 \\ 386950370 & 162876845 & 67844842 & 27929582 & 11345952 & 4540152 & 1785854 & 688841 & 259838 & 95567 & 34170 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{10} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Now we solve the above to obtain the vectors for $Q(t)$. One way to do this is by using the software Maple with the *LinearAlgebra* package. In the current example we have a unique solution for $Q(t)$:

$$\begin{bmatrix} 1 & -14 & 87 & -316 & 743 & -1182 & 1289 & -952 & 456 & -128 & 16 \end{bmatrix}^{\mathsf{T}}.$$

Equation (2.3) is then given by

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 42 & 14 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 131 & 42 & 14 & 5 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 414 & 131 & 42 & 14 & 5 & 2 & 1 & 1 & 0 & 0 & 0 \\ 1298 & 414 & 131 & 42 & 14 & 5 & 2 & 1 & 1 & 0 & 0 \\ 3980 & 1298 & 414 & 131 & 42 & 14 & 5 & 2 & 1 & 1 & 0 \\ 11848 & 3980 & 1298 & 414 & 131 & 42 & 14 & 5 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -14 \\ 87 \\ -316 \\ 743 \\ -1182 \\ 1289 \\ -952 \\ 456 \\ -128 \\ 16 \end{bmatrix}$$

which we solve to find the vectors for $P(t)$:

$$\begin{bmatrix} 1 & -13 & 75 & -252 & 545 & -790 & 774 & -502 & 201 & -43 & 5 \end{bmatrix}^{\intercal}.$$

Finally, substituting the coefficients for $P(t)$ and $Q(t)$ and factoring gives the following generating function for $\mathrm{Av}(321, 132564)$:

$$\frac{5t^{10} - 43t^9 + 201t^8 - 502t^7 + 774t^6 - 790t^5 + 545t^4 - 252t^3 + 75t^2 - 13t + 1}{(2t-1)^4(t-1)^6} \tag{2.4}$$

To check its validity, we expand it as a power series which gives

$$\begin{aligned} &1 + t + 2t^2 + 5t^3 + 14t^4 + 42t^5 + 131t^6 + 414t^7 + 1298t^8 + 3980t^9 + 11848t^{10} \\ &+ 34170t^{11} + 95567t^{12} + 259838t^{13} + 688841t^{14} + 1785854t^{15} + 4540152t^{16} \\ &+ 11345952t^{17} + 27929582t^{18} + 67844842t^{19} + 162876845t^{20} + 386950370t^{21} \\ &+ 910721351t^{22} + O(t^{23}) \end{aligned}$$

This completely matches the 22 terms computed from DFS earlier, in particular the coefficient for t^{22} was correctly predicted by the computed rational function, which shows that the generating function given by Equation 2.4 is a genuine guess for the class $\mathrm{Av}(321, 132564)$.

Using the method described above, we have computed the generating function guesses of $\mathrm{Av}(321, \pi)$ for every π of length 5, as well as for most π of length 6. The list of conjectured generating functions, along with the first few terms of the enumerating sequence, is shown in Appendix A. For certain cases where $|\pi| = 6$ we are unable to compute the generating functions due to the large amount of terms needed to solve Equation (2.2) - some require at least 27 terms, which may not be feasible under the current processing power.

2.5 Observations and concluding remarks

We now have a big list of generating functions of the form $\mathrm{Av}(321, \pi)$, where $|\pi| \leqslant 6$. The last section addressed the cases where $|\pi| = 5, 6$ by making genuine guesses. For the classes with $|\pi| = 4$, we refer the reader to the enumeration by Atkinson [Atk99], or West [Wes96], and for $|\pi| = 3$, Simion and Schmidt [SS85].

With the data in hand, a natural question would be whether we can find any pattern among the listed generating functions. In particular, if we can classify classes with certain structural properties by generating functions of a certain form, we may be able to derive the generating function (or at least part of it) of more classes of the same type without having to enumerate them.

Here we propose two observations which we found from the data (the first happens to be an alternative form of an established theorem), along with open problems that arise from the observations. We believe these findings may be useful for future research on characterising possible enumeration functions of 321-avoiding classes.

Theorem 2.2. *The generating function of* $\mathrm{Av}(321, \pi)$*, where* π *is a pattern of length* n *and has the form* $n\,1\,2\,\ldots\,n-1$*, is*

$$\frac{(-1)^{n-1} g_{n-1}(t)}{g_n(t)}$$

where $g_n(t)$ *is defined by the following recurrence relation:*

$$g_n(t) = \begin{cases} 1 & \text{if } n = 1 \\ t - 1 & \text{if } n = 2 \\ (-1)^{n-1} g_{n-1}(t) + t g_{n-2}(t) & \text{if } n > 2 \end{cases}$$

Table 2.4 provides a list of every permutation of length 4, 5 and 6 satisfying Theorem 2.2. In addition, all non-trivial Wilf equivalences have been indicated by the different colours in each Wilf equivalent class, and two permutations are of the same colour if and only if they can be obtained from each other via symmetry.

As it happens, Theorem 2.2 above is in fact a different form of the same theorem as Theorem 3.1 of [CW99], in which Chow and West expressed the generating function of the same class in terms of Chebyshev polynomials. Mansour and Vainshtein generalised the theorem of Chow and West in [MV00] by showing that for every 2-layered pattern π, $\mathrm{Av}(321, \pi)$ can be enumerated by the same formula - here a permutation is said to be *p-layered* if it consists of the disjoint union of p substrings (the layers) so that the entries decrease within each layer, and increase between the layers. Their generalisation, however, still did not include every permutation to which the formula applies. For example, for $n = 6$, the only 2-layered π are 234561, 345612, 456123, and their symmetries as shown in Table 2.4. While for classes with $\pi =$ 245613, 246135, 351624, 356124 and their symmetries, the theorem by Mansour and Vainshtein does not apply, and it remains open as to why these classes appear to be Wilf equivalent with the 2-layered classes.

The next conjecture provides a criterion for a class to have GF with denominator of the form $(2t-1)^a(t-1)^b$.

Conjecture 2.3. *Let π be a pattern and $\sigma_1 \oplus \sigma_2 \oplus \cdots \oplus \sigma_k$ be its sum decomposition into sum-indecomposable components. Then the denominator of the generating function of $\mathrm{Av}(321, \pi)$ has the form $(2t-1)^a(t-1)^b$ if each σ_i is of length at most 3.*

A special subcase of Conjecture 2.3 is when $\pi = \sigma_1 \oplus \cdots \oplus 21 \oplus \cdots \oplus \sigma_k$, where the denominator of the generating function of $\mathrm{Av}(321, \pi)$ has the form $(t-1)^a$ if each σ_i is of length 1. This is precisely the case when the class has polynomial enumeration, and was established by Albert, Atkinson, and Brignall in [AAB07].

As no work has yet been done on proving or disproving Conjecture 2.3, the polynomial classes remain the most complicated 321-avoiding classes with a partially known

π	**rational generating function of** $\mathrm{Av}(321, \pi)$
2341 4123 2413 3142 3412	$\frac{-(2t-1)}{t^2-3t+1}$
23451 51234 24513 34152 35124 41523 34512 45123	$\frac{t^2-3t+1}{(3t-1)(t-1)}$
234561 612345 245613 345162 461235 516234 246135 415263 345612 561234 351624 356124 451623 456123	$\frac{-(3t-1)(t-1)}{t^3-6t^2+5t-1}$

Table 2.4: All permutations of length 4, 5, and 6 that satisfy Theorem 2.2

structure. As a result, in the next chapter we will study such classes in detail, and eventually aim to enumerate polynomial classes of the form $\mathrm{Av}(321, \pi)$ using a structural, constructive approach.

Chapter 3

Permutation classes of polynomial growth

In this chapter we examine enumerating polynomial classes from a structural description. In particular, given any polynomial class $\mathcal{C}$ along with its associated grid class (in the form of a set $\tilde{G}$ of peg permutations), Homberger and Vatter proposed in [HV15] an algorithm which computes the generating function of the class. Here we give an overview of the work by Homberger and Vatter in the first two sections of the chapter. In the two sections that follow, we show that for a polynomial class of the form $\mathrm{Av}(321, \beta)$, we can compute its associating grid class as well as its generating function when only the basis is known. An algorithm similar to that in [HV15] is provided for enumerating such classes, along with an example showing how the algorithm works in practice.

3.1 Introduction

A permutation class $\mathcal{C}$ is said to have *polynomial growth*, or is called a *polynomial class*, if the number of permutations of each length n in $\mathcal{C}$ is bounded by

$$\mathcal{C}_n \leqslant An^d$$

for some constants A, d. It is known by Kaiser and Klazar [KK03] that a permutation class of polynomial growth has an eventually polynomial enumeration.

There have been several works on defining polynomial classes from a structural description. Huczynska and Vatter showed in [HV06] that there are some specific polynomial classes given by taking specific permutations and allowing substitution of monotone intervals, and every polynomial class is a subclass of one of these. Hence we can describe a polynomial class by a finite partition of monotone intervals. In the paper [HV15] by Homberger and Vatter, the partitioning into monotone intervals was represented by the notion of peg permutations. The paper then provided a method to enumerate a polynomial class when such a partitioning is given.

Albert, Atkinson, and Brignall [AAB07] provided an alternative condition for which $\mathrm{Av}(B)$ has polynomial growth, which is based on finding all of the 10 types of patterns described in the paper, among the permutations in the basis B. In particular, for any class with two basis elements, i.e. $|B| = 2$, we have the following restrictions on its basis:

Theorem 3.1 ([AAB07], Theorem 4)**.** *The class $X = \mathrm{Av}(\alpha, \beta)$ has polynomial growth if and only if (up to symmetry and exchange of α with β) we have one of the following:*

1. *α is increasing and β is decreasing,*
2. *α is increasing and β is almost decreasing in the sense that $\beta = 1^k \ominus 12 \ominus 1^l$, $k, l \geqslant 0$ with exactly one non-decreasing skew sum component of size 2.*

For our interest, Theorem 3.1 shows that any non-finite class of the form $\mathrm{Av}(321, \beta)$ is polynomial if and only if $\beta = 1^k \oplus 21 \oplus 1^l$. In the second half of the Chapter, we study polynomial classes of the form $\mathrm{Av}(321, \beta)$, and show that given any basis β, we can derive the partitioning into monotone intervals of the class, then enumerate it based on the method given by Homberger and Vatter.

3.2 General structure of polynomial classes in terms of peg permutations

In this section we formalize the notion of the structure of polynomial classes in the way decribed in [HV15]. As mentioned in the introductory section, a polynomial class can be described by a finite partition of monotone intervals. One way of representing a polynomial class is by using the grid classes of peg permutations, which is illustrated later in Theorem 3.2. Here the partitioning into monotone intervals is represented by

the notion of peg permutations, where each element of the peg permutation can be inflated by such an interval.

A *peg permutation* is a permutation where each element is decorated with a $+$, $-$, or $\bullet$. We denote peg permutations by symbols such as $\tilde{\rho}$, where ρ is the underlying (non-pegged) permutation. The *grid class* of a peg permutation $\tilde{\rho}$, denoted $\mathrm{Grid}(\tilde{\rho})$, is the set of all permutations that may be obtained by inflating ρ by monotone intervals of type determined by the signs of $\tilde{\rho}$: each element of $\tilde{\rho}$ decorated with a $+$ (resp. $-$) symbol can be replaced by an arbitrary increasing (resp. decreasing) interval, while those decorated with a dot may only be replaced by a single point. Note that here we allow replacement by the empty permutation in all of the above cases unless stated otherwise.

It is then clear that every permutation in the grid class $\mathrm{Grid}(\tilde{\rho})$ can be partitioned into monotone intervals compatible with $\tilde{\rho}$, we call this a *$\tilde{\rho}$-partition* of π. Given a set $\tilde{G}$ of peg permutations, the union of their grid classes is denoted by

$$\mathrm{Grid}(\tilde{G}) = \bigcup_{\tilde{\rho}\in\tilde{G}} \mathrm{Grid}(\tilde{\rho}).$$

The following theorem, which is a combination of the results of Huczynska and Vatter [HV06] and Albert, Atkinson, Bouvel, Ruškuc and Vatter [AAB+11], describes the structure of a polynomial class in terms of grid classes of peg permutations.

Theorem 3.2. *A permutation class $\mathcal{C}$ has polynomial enumeration if and only if $\mathcal{C} = \mathrm{Grid}(\tilde{G})$ for a finite set $\tilde{G}$ of peg permutations.*

Because we only replace peg permutations by monotone intervals, we can specify these intervals by vectors of non-negative integers, where $\vec{v}(i)$ gives the number of points the i^{th} element of $\tilde{\rho}$ gets inflated by. Then any permutation in the class $\mathcal{C}$ can be obtained from $\tilde{\rho}[\vec{v}]$ for $\tilde{\rho} \in \tilde{G}$ and some $\vec{v}$, where $\vec{v} \in \mathbb{N}^m$ is a vector of length m, and $\mathbb{N}$ denotes the non-negative integers. We denote the *weight* of $\vec{v}$ as $\|\vec{v}\| = \sum \vec{v}(i)$.

For example,

$$6\,321\,7\,45 = 3^{\bullet}1^{-}4^{\bullet}2^{+}[\langle 1,3,1,2\rangle].$$

Let $\tilde{\rho}$ be a peg permutation of length m and $\vec{v} \in \mathbb{N}^m$. If $\tilde{\rho}(i)$ is dotted, we must have $\vec{v}(i) \leqslant 1$. Thus if $\tilde{\rho}$ is of length m, we can write

$$\mathrm{Grid}(\tilde{\rho}) = \{\tilde{\rho}[\vec{v}] : \vec{v} \in \mathbb{N}^m \text{ which satisfy } \vec{v}(i) \leqslant 1 \text{ for all } i \text{ such that } \tilde{\rho}(i) \text{ is dotted}\}.$$

Note again that here we allowed inflation by the empty permutation for the definition of grid classes. In our later algorithm and structural theorem (Theorem 3.3), however, we want every permutation in $\mathcal{C}$ to be obtainable by inflating a unique irreducible base permutation, hence we only consider inflating an entry by one or more elements. As such, we insist on inflating $\tilde{\rho}$ by vectors which *fill* them, this means each component of the vector corresponding to a dotted element of $\tilde{\rho}$ equals 1, and is otherwise at least 2.

Given a set $\mathcal{V} \subseteq \mathbb{P}^m$ of vectors which fill $\tilde{\rho}$, we define

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

We also extend the notion of containment and avoidance to vectors. Given the vectors $\vec{v}$ and $\vec{w}$ in $\mathbb{N}^m$, we say that $\vec{v}$ is *contained* in $\vec{w}$ if $\vec{v}(i) \leqslant \vec{w}(i)$ for all indices i, and $\tilde{w}$ avoids $\vec{v}$ if $\vec{v}$ is not contained in $\vec{w}$. We then define the *minimal filling vector* $\vec{m}$ of a peg permutation $\tilde{\rho}$ as the smallest vector by containment that fills $\tilde{\rho}$. It is easy to see that if $\vec{v} \leqslant \vec{w}$ then $\tilde{\rho}[\vec{v}] \leqslant \tilde{\rho}[\vec{w}]$, assuming both inflations are defined.

Because our order on vectors is a partial order, we define the *downsets* (resp. upsets) of vectors to be the sets closed downward (resp. upward) under containment. The intersection of a downset and an upset is referred to as a *convex set*.

Given a peg permutation $\tilde{\rho}$, and any set $\mathcal{V}$ of vectors that fill $\tilde{\rho}$, we note that $\mathcal{V}$ is a convex set of filling vectors of $\tilde{\rho}$: the downset component of $\mathcal{V}$ consists of vectors which do not contain an entry larger than 1 corresponding to a dotted entry of $\tilde{\rho}$, while the upset component consists of vectors which contain the minimal filling vector $\vec{m}$ of $\tilde{\rho}$, in this case $\vec{m}(i) = 1$ if $\tilde{\rho}(i)$ is dotted and $\vec{m}(i) = 2$ if $\tilde{\rho}(i)$ is signed; then we see that a vector is in $\mathcal{V}$ if and only if it satisfies both conditions described above.

The following theorem is a key ingredient in the practical enumeration of any polynomial class $\mathcal{C}$.

Theorem 3.3. *For every polynomial permutation class $\mathcal{C}$ there is a finite set $\tilde{H}$ of peg permutations, each associated with its own convex set $\mathcal{V}_{\tilde{\rho}}$ of filling vectors, such that $\mathcal{C}$ can be written as the disjoint union*

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

Homberger and Vatter [HV15] proved Theorem 3.3 by giving and justifying an algorithm to compute the set $\tilde{H}$ and the associated convex sets $\mathcal{V}_{\tilde{\rho}}$ for each $\tilde{\rho} \in \tilde{H}$, which we

show in the next section. The enumeration of the class is then reduced to enumerating a finite number of convex sets of vectors.

3.3 The Homberger-Vatter Algorithm

We now describe the algorithm proposed by Homberger and Vatter, which given the set $\tilde{G}$ of peg permutations, computes the set $\tilde{H}$ of peg permutations and the associated convex sets of vectors as specified in the statement of Theorem 3.3. The set $\tilde{H}$ is then enumerated to compute the generating function of the class $\mathcal{C} = \mathrm{Grid}(\tilde{G})$. This process is divided into five steps which we summarise below:

1. Completion: the input set $\tilde{G}$ is completed by replacing it with its downward closure $\tilde{H}$
2. Compacting: the resulting set is then compacted, by removing each element of $\tilde{G}$ that has an interval isomorphic to one of those listed in Proposition 3.5
3. Cleaning: the resulting set is then cleaned, by removing each element of $\tilde{G}$ that has an interval isomorphic to $1^\bullet 2^\bullet$ and $2^\bullet 1^\bullet$
4. Combination: convex sets associated to the same peg permutation are combined, to ensure the sets $\tilde{\rho}[\mathcal{V}_{\tilde{\rho}}]$ are disjoint as promised by Theorem 3.3
5. Enumeration: the generating function is finally computed from the set $\tilde{H}$ obtained after the first four steps

We will explain each of the steps in their own subsections below. Alongside our description we also enumerate as an example the class $\mathrm{Grid}(1^- 2^+)$. (this is the same example as the one illustrated in [HV15])

Here we define a partial order on peg permutations. Given peg permutations $\tilde{\tau}$ and $\tilde{\rho}$ of length k and n respectively, $\tilde{\tau} \leqslant \tilde{\rho}$ if there are indices $1 \leqslant i_1 < 1_2 < \cdots < i_k \leqslant n$ such that $\rho(i_1)\rho(i_2)\cdots\rho(i_k)$ is order isomorphic to τ and for each j, $\tilde{\tau}(j)$ is decorated with a

$$\begin{cases} + \text{ or } \bullet & \text{if } \tilde{\rho}(i_j) \text{ is decorated with a } +, \\ - \text{ or } \bullet & \text{if } \tilde{\rho}(i_j) \text{ is decorated with a } -, \textit{or} \\ \bullet & \text{if } \tilde{\rho}(i_j) \text{ is dotted.} \end{cases}$$

This definition implies one can obtain a smaller element in *peg permutation order* by deleting entries or replacing + or − signs by •.

Step 1. Completion

We say that the set $\tilde{G}$ of peg permutations is *complete* if every $\pi \in \mathrm{Grid}(\tilde{G})$ fills some $\tilde{\rho} \in \tilde{G}$. The following proposition shows how we can turn an arbitrary set of peg permutations into a complete set.

Proposition 3.4. *Every downset in the peg permutation order is complete.*

Given an input set $\tilde{G}$ of peg permutations, the completion step involves taking the downward closure $\tilde{H}$ of $\tilde{G}$, which can be obtained by deleting entries or replacing + or − signs by • in every peg permutations of $\tilde{G}$. Then $\tilde{H}$ is a complete set of peg permutations by Proposition 3.4, and every permutation in $\mathrm{Grid}(\tilde{H})$ fills some member of $\tilde{H}$.

In our example $\tilde{G} = \{1^-2^+\}$, the completion step takes the downward closure of $\tilde{G}$ which is

$$\{1^\bullet, 1^\bullet 2^\bullet, 1^+, 1^-, 1^\bullet 2^+, 1^- 2^\bullet, 1^- 2^+\}.$$

Step 2. Compacting

A peg permutation $\tilde{\rho}$ is *compact* if $\mathrm{Grid}(\tilde{\tau}) \subsetneq \mathrm{Grid}(\tilde{\rho})$ for all $\tilde{\tau} < \tilde{\rho}$. For example, $1^+2^\bullet$ is not compact since $1^+ < 1^+2^\bullet$ and $\mathrm{Grid}(1^+2^\bullet) = \mathrm{Grid}(1^+)$. On the other hand, both 1^+ and $1^\bullet 2^\bullet$ are compact.

Proposition 3.5 (Proposition 2.3 of [HV15])**.** *For a peg permutation $\tilde{\rho}$, the following conditions are equivalent:*

1. *$\tilde{\rho}$ is compact,*
2. *$\tilde{\rho}$ does not have an interval order isomorphic to 1^+2^+, $1^+2^\bullet$, $1^\bullet 2^+$, or symmetrically, to 2^-1^-, $2^-1^\bullet$, $2^\bullet 1^-$, and*
3. *every permutation which fills $\tilde{\rho}$ has a unique $\tilde{\rho}$-partition.*

Proposition 3.5 shows us that we can easily identify non-compact elements of $\tilde{G}$, by checking each $\tilde{\rho} \in \tilde{G}$ whether it contains an interval isomorphic to one of those listed

in the Proposition. Now we say the set $\tilde{G}$ of peg permutations is compact if every peg permutation in $\tilde{G}$ is compact. Note that if $\tilde{G}$ is a downset and $\tilde{\rho} \in \tilde{G}$ is not compact then there is a $\tilde{\tau} \in \tilde{G}$ such that $\mathrm{Grid}(\tilde{\tau}) = \mathrm{Grid}(\tilde{\rho})$. This implies the following result.

Proposition 3.6 (Proposition 2.4 of [HV15])**.** *Let $\tilde{G}$ be a downset of peg permutations and $\tilde{H}$ the result of removing all non-compact peg permutations from $\tilde{G}$. Then* $\mathrm{Grid}(\tilde{G}) = \mathrm{Grid}(\tilde{H})$.

Proposition 3.6 says we can obtain a compact set from $\tilde{G}$ by simply removing all its non-compact elements. Combining it with Proposition 3.5 which finds such elements and we have the method which compacts any complete set $\tilde{G}$.

In our running example, the only non-compact peg permutation in the complete set is $1^\bullet 2^+$, which we remove from $\tilde{G}$ in the compacting step:

$$\{1^\bullet, 1^\bullet 2^\bullet, 1^+, 1^-, \cancel{1^\bullet 2^+}, 1^- 2^\bullet, 1^- 2^+\}.$$

Step 3. Cleaning

The cleaning step addresses the problem that some entries of a peg permutation $\tilde{\rho}$ may only be inflated by a finite number of elements. We say that a compact peg permutation $\tilde{\rho}$ is *clean* if $\mathrm{Grid}(\tilde{\rho}) \not\subseteq \mathrm{Grid}(\tilde{\tau})$ for any shorter peg permutation $\tilde{\tau}$, and the set $\tilde{G}$ of peg permutations is clean if each of its elements is clean.

Proposition 3.7. *The compact peg permutation $\tilde{\rho}$ is clean if and only if it does not have an interval order isomorphic to $1^\bullet 2^\bullet$ or $2^\bullet 1^\bullet$.*

As an example, $2^\bullet 3^\bullet 4^\bullet 1^+$ is not clean as $\mathrm{Grid}(2^\bullet 3^\bullet 4^\bullet 1^\bullet) \subseteq \mathrm{Grid}(2^+ 1^\bullet)$. Note that since 2341 fills both $2^\bullet 3^\bullet 4^\bullet 1^\bullet$ and $2^+ 1^\bullet$, we will have redundancy if both of these are in the set $\tilde{G}$. However, it may happen that $2^\bullet 3^\bullet 4^\bullet 1^\bullet \in \tilde{G}$ but $2^+ 1^\bullet \notin \tilde{G}$, and in that case we cannot remove the redundancy by simply deleting $2^\bullet 3^\bullet 4^\bullet 1^\bullet$, or adding $2^+ 1^\bullet$ from $\tilde{G}$. Instead, we clean the set by contracting each unclean $\tilde{\rho} \in \tilde{G}$ and restricting convex sets of the resulting permutation, in the following way:

- If $\tilde{\rho}$ is clean, leave it alone.
- Otherwise, contract the maximal intervals in $\tilde{\rho}$ of the form $1^\bullet 2^\bullet \dots k^\bullet$ (resp., $k^\bullet \dots 2^\bullet 1^\bullet$), $k \geqslant 2$ to 1^+ (resp. 1^-), and call the new peg permutation $\tilde{\tau}$. To $\tilde{\tau}$ we associate the convex set of vectors which fill $\tilde{\tau}$ and for each entry i, if the i^{th}

entry of $\tilde{\tau}$ is the result of contracting k dotted entries, then the vectors of $\mathcal{V}_{\tilde{\tau}}$ may not have their i^{th} component greater than k.

In our running example, the only unclean peg permutation in the compact set is $1^\bullet 2^\bullet$, which we replace by 1^+ and associate with the convex set of vectors $\{\langle 2 \rangle\}$. Thus we have

$$\{1^\bullet, \cancel{1^\bullet 2^\bullet}, 1^+, 1^-, 1^- 2^\bullet, 1^- 2^+\}.$$

Step 4. Combination

The combination step fixes the potential problem of having multiple convex sets associated to each peg permutation, which may result from the previous cleaning step. For a pair of vectors $\vec{v}$ and $\vec{w}$, we define their meet and join by

$$\begin{aligned}\vec{v} \wedge \vec{w} &= (\min\{v(1), w(1)\}, \ldots, \min\{v(m), w(m)\}),\\ \vec{v} \vee \vec{w} &= (\max\{v(1), w(1)\}, \ldots, \max\{v(m), w(m)\}).\end{aligned}$$

The next Proposition shows how we can compute the union and intersection of arbitrary vector posets.

Proposition 3.8. *If $\mathcal{V}, \mathcal{W} \subseteq \mathbb{P}^m$ be downsets with bases $B_\mathcal{V}$ and $B_\mathcal{W}$, respectively, then $\mathcal{V} \cap \mathcal{W}$ and $\mathcal{V} \cup \mathcal{W}$ are also downsets. Further, the basis of $\mathcal{V} \cap \mathcal{W}$ is given by the minimal elements of the set $B_\mathcal{V} \cup B_\mathcal{W}$, and the basis of $\mathcal{V} \cup \mathcal{W}$ is given by the minimal elements of the set $\{\vec{v} \vee \vec{w} : \vec{v} \in B_\mathcal{V}$ and $\vec{w} \in B_\mathcal{W}\}$.*

By Proposition 3.8, we can combine multiple convex sets associated with the same permutation by taking their union, after which we get a unique convex set associated to each peg permutation.

We have now what is required to prove Theorem 3.3. The steps 2-4 established that every permutation $\pi \in \text{Grid}(\tilde{G})$ fills a unique clean and compact peg permutation $\tilde{\rho}$. Thus there is a unique vector $\vec{v} \in \mathcal{V}_{\tilde{\rho}}$ such that $\pi = \tilde{\rho}[\vec{v}]$. It then follows that

$$\mathcal{C} = \text{Grid}(\tilde{G}) = \biguplus_{\tilde{\rho} \in \tilde{G}} \tilde{\rho}[\mathcal{V}_{\tilde{\rho}}].$$

In our example, the convex sets associated to the peg permutations are respectively

$$\begin{aligned}
\mathcal{V}_{1^\bullet} &= \{\langle 1\rangle\},\\
\mathcal{V}_{1^+} &= \mathbb{P}\backslash\{\langle 1\rangle\},\\
\mathcal{V}_{1^-} &= \mathbb{P}\backslash\{\langle 1\rangle\},\\
\mathcal{V}_{1^-2^\bullet} &= (\mathbb{P}\times\{1\})\backslash\{\langle 1,1\rangle\},\\
\mathcal{V}_{1^-2^+} &= \mathbb{P}^2\backslash(\mathbb{P}\times\{1\}\cup\{1\}\times\mathbb{P}).
\end{aligned}$$

Step 5. Enumeration

We summarise the above four steps in Algorithm 4, which computes the set $\tilde{H}$ required in Theorem 3.3.

To compute the generating function from $\tilde{H}$, first note that for any vector $\vec{w}\in\mathbb{P}^m$, the generating function for vectors $\vec{v}\in\mathbb{P}^m$ which satisfy $\vec{v}\geqslant\vec{w}$ is

$$\frac{x^{\|\vec{w}\|}}{(1-x)^m}.$$

Let $\mathcal{V}$ be a downset of vectors of length m with finite basis B. Then by the Principle of Inclusion-Exclusion, the generating function for vectors in $\mathcal{V}$ is

$$\sum_{B\subseteq B_{\mathcal{V}}}(-1)^{|B|}\frac{x^{\|\bigvee B\|}}{(1-x)^m}.$$

Since the complement of an upset is a downset, the above formula also works for convex sets $\mathcal{V}$. Summing up the generating function for each convex set then gives the generating function for the class.

In our example, the generating functions of each of the convex sets are respectively

$$x,\ \frac{x^2}{1-x},\ \frac{x^2}{1-x},\ \frac{x^3}{1-x},\ \frac{x^4}{(1-x)^2}.$$

Summing these gives the generating function for the class,

$$\sum_{n\geqslant 1}\left|\mathrm{Grid}_n(1^-2^+)\right|x^n=\frac{x}{(1-x)^2}=\sum_{n\geqslant 1}nx^n.$$

Algorithm 4: Summary of the Homberger-Vatter algorithm

Input : Set $\tilde{G}$ of peg permutations

Output: A set $\tilde{H}$ of peg permutations, each associated with a convex set $\mathcal{V}_{\tilde{\rho}}$ of vectors so that $\mathrm{Grid}(\tilde{G})$ is the disjoint union $\uplus_{\tilde{\rho}\in\tilde{H}}\tilde{\rho}[\mathcal{V}_{\tilde{\rho}}]$.

// Complete $\tilde{G}$

for $\tilde{\rho} \in \tilde{G}$ **do**

  Add to $\tilde{G}$ all peg permutations which are contained in $\tilde{\rho}$ in the peg permutation order

// Compact $\tilde{G}$

for $\tilde{\rho} \in \tilde{G}$ **do**

  if $\tilde{\rho}$ *contains intervals of the form* 1^+2^+, $1^\bullet2^+$, $1^+2^\bullet$, *or their symmetries* **then**

    remove $\tilde{\rho}$ from $\tilde{G}$

// Clean and combine $\tilde{G}$

Initialize the set $\tilde{H}$, which will contain pairs $(\tilde{\rho}, \mathcal{V}_{\tilde{\rho}})$ of peg permutations associated with convex sets of vectors

for $\tilde{\rho} \in \tilde{G}$ **do**

  if $\tilde{\rho}$ *contains intervals of the form* $1^\bullet2^\bullet$ *or* $2^\bullet1^\bullet$ **then**

    Let $\tilde{\gamma}$ denote the cleaned $\tilde{\rho}$ and define $\mathcal{V}$ to be the set of integer vectors for which $\{\tilde{\gamma}[\vec{v}] : \vec{v} \in \mathcal{V}\} = \{\tilde{\rho}[\vec{v}] : \vec{v} \text{ fills } \tilde{\rho}\}$

  else

    Let $\tilde{\gamma} = \tilde{\rho}$ and $\mathcal{V} = \{\vec{v} : \vec{v} \text{ fills } \tilde{\rho}\}$

  if $(\tilde{\gamma}, \mathcal{W}) \in \tilde{H}$ *for some* $\mathcal{W}$ **then**

    Replace the element $(\tilde{\gamma}, \mathcal{W})$ with $(\tilde{\gamma}, \mathcal{W} \cup \mathcal{V})$

  else

    Add $(\tilde{\gamma}, \mathcal{V})$ to $\tilde{H}$

return $\tilde{H}$

3.4 Enumerating polynomial classes of the form $\mathrm{Av}(321, \beta)$

In this section, we specialise and adapt the preceding discussion to enumerate polynomial classes of the form $\mathrm{Av}(321, \beta)$. Albert, Atkinson, and Brignall proved in [AAB07] that any 321-avoiding class $\mathrm{Av}(321, \beta)$ is polynomial if and only if β is almost increasing in the sense that $\beta = 1^k \oplus 21 \oplus 1^l$ for some integers k, l. Due to symmetry, in this thesis we assume without loss of generality that $k \geqslant l$.

The section is divided into two parts. In the first part we determine the structure of the class $\mathcal{C} = \mathrm{Av}(321, \beta)$ and find an associated set $\tilde{G}$ of peg permutations which satisfies $\mathcal{C} = \mathrm{Grid}(\tilde{G})$ as stated in Theorem 3.2. In the second part we describe an algorithm similar to the Homberger-Vatter algorithm presented in Section 3.3, which computes from $\tilde{G}$ the set $\tilde{H}$ required in the structural theorem (Theorem 3.3) of the previous section.

3.4.1 Structure of maximal plus irreducibles

Let $\mathcal{C}_{k,l}$ denote the polynomial class $\mathrm{Av}(321, \beta)$ where $\beta = 1^k \oplus 21 \oplus 1^l$, $k \geqslant l$. We define the *maximal plus irreducibles* of $\mathcal{C}_{k,l}$ as follows:

Definition 3.1. *The maximal plus irreducibles (or* MPI*) of the class $\mathcal{C}_{k,l}$ are the largest plus irreducible permutations in $\mathcal{C}$ by containment. i.e. they are not contained in any other plus irreducibles of $\mathcal{C}$.*

Lemma 6 of [AAB07] showed that in $\mathcal{C}_{k,l}$ such largest plus irreducible permutations exist and have finite bounded size. Denote by $\mathrm{MPI}(\mathcal{C}_{k,l})$ the set of maximal plus irreducibles in the class $\mathcal{C}_{k,l}$, the following proposition shows that the elements of $\mathrm{MPI}(\mathcal{C}_{k,l})$ are precisely the underlying permutations of $\tilde{\rho} \in \tilde{G}$.

Proposition 3.9. *Let $\tilde{G}$ be the smallest set such that $\mathcal{C}_{k,l} = \mathrm{Grid}(\tilde{G})$. Then the maximal plus irreducibles of $\mathcal{C}_{k,l}$ are precisely the underlying permutations of $\tilde{\rho} \in \tilde{G}$.*

Proof. First note that every plus irreducible permutation q that is not maximal (i.e. is contained in some $\rho \in \mathrm{MPI}(\mathcal{C})$) can be obtained from ρ by removing (which is equivalent to inflating by an empty permutation) all entries of ρ not involved in the embedding from q to ρ. In addition, every plus reducible permutation can be obtained from inflating some plus irreducible. Thus we can obtain every permutation in $\mathcal{C}_{k,l}$

by inflating the maximal plus irreducibles, which implies $\mathcal{C}_{k,l} = \mathrm{Grid}(\tilde{G})$, with the maximal plus irreducibles of $\mathcal{C}_{k,l}$ being the underlying permutations of $\tilde{\rho} \in \tilde{G}$. $\square$

We thus call $\tilde{\rho} \in \tilde{G}$ a *pegged maximal plus irreducible* (or pMPI) of $\mathcal{C}_{k,l}$, and $\tilde{G}$ the *set of pegged maximal plus irreducibles*.

We shall compute the set $\tilde{G}$ from $\mathrm{MPI}(\mathcal{C})$ with Lemma 3.10 and Theorem 3.13 below. Given a 321 avoiding permutation π, we call all elements that form a 2 in a copy of 21 *upper elements*, and the remaining ones *lower elements* (note that the fluid elements defined in Chapter 1 are considered here also as lower elements), both of which form increasing sequences. We index the upper elements from left to right as $u_1, u_2, \ldots$, and the lower elements as $\ell_1, \ell_2, \ldots$.

Lemma 3.10. *Let ρ be a maximal plus irreducible of $\mathcal{C}_{k,l}$. Then the following holds for each $1 \leqslant i \leqslant k-1$:*

(Property A_i) *the i^{th} upper element u_i is above and to the left of the i^{th} lower element ℓ_i of ρ.*

To prove Lemma 3.10, we show that if Property A_i does not hold up to $k-1$, then ρ can always be extended into a larger plus irreducible ρ' (and is thus not maximal). Here the following observations would be useful for checking β avoidance in ρ'.

Observation 3.1. *If x is an element of $\rho \in \mathrm{Av}(321)$, and the longest increasing subsequence in ρ ending at x has length $t \leqslant k$, then x can only occur as one of the first t elements of an occurrence of β.*

Observation 3.2. *If, for some $t \leqslant k$, Property A_i holds up to $t-1$ for $\rho \in \mathrm{Av}(321, \beta)$, then any increasing sequence of u's and l's among first $t-1$ must increase subscripts.*

This is easy to see since any u_s (resp. ℓ_s) above and after ℓ_i (resp. u_i) must have subscript $s > i$ for $i \leqslant t-1$.

Observation 3.3. *If $\rho \in \mathrm{Av}(321, \beta)$ and $\rho' = \rho \cup \{x\}$, where $x \in \rho' \in \mathrm{Av}(321)$ and the longest increasing subsequence in ρ' ending at x has length $t \leqslant k$, then ρ' avoids β if and only if the area above and to the right of x avoids $1^{k-t} \oplus 21 \oplus 1^l$.*

Observation 3.3 follows directly from Observation 3.1 since ρ' must have used x if ρ' were to contain β. We now prove Lemma 3.10 below.

Proof of Lemma 3.10. We prove by induction on i that, if a plus irreducible permuta-

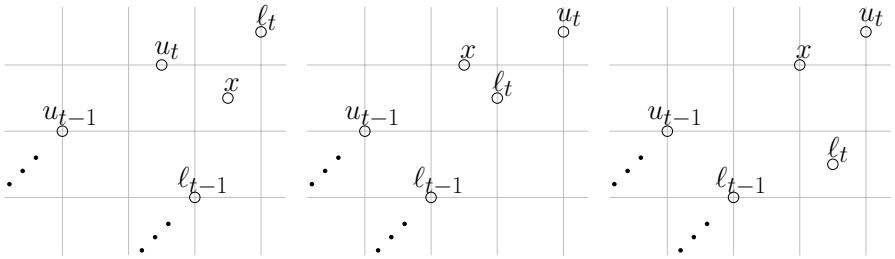


Figure 3.1: Extending ρ to ρ' in Lemma 3.10 by inserting x. Case 1 (left). Case 2 (centre). Case 3 (right).

tion $\rho \in \mathrm{Av}(321, \beta)$ satisfies Property A_1 up to A_{i-1} but not A_i, then ρ is not maximal as we can extend it by a single element to a larger plus irreducible ρ'. Throughout the remainder of the proof, ρ' will denote a plus irreducible extension of ρ by a single element x belonging to $\mathcal{C}_{k,l}$ which establishes that ρ is not maximal. When we specify these extensions it will always be clear that ρ' does not contain a 321 and is plus-irreducible, so the only issue will be to prove that it avoids β. Since ρ avoids β, any possible copy of β in ρ' would necessarily use x.

$(i = 1)$ Suppose in $\rho_1 \in \mathrm{Av}(321, \beta)$, u_1 does not precede and lie above ℓ_1, then $\rho_1 = 1e_2e_3\ldots$ and $\ell_1 = 1$. We extend ρ_1 by inserting an element $1^{+\epsilon}$ just larger than $\ell_1 = 1$ and immediately before ℓ_1 to create $\rho_1' = 1^{+\epsilon}1e_2e_3\ldots$. By construction it is clear that ρ_1' is plus irreducible. Now if ρ' contains β it would have to use the $1^{+\epsilon}$, but then ρ would have a β using 1 instead.

$(i = t)$ Now suppose for $i = t \leqslant k-1$, Property A_i holds up to $t-1$ and u_t does not precede and lie above ℓ_t. We consider the following three cases:

Case 1 u_t precedes but lies below ℓ_t.

> **Extension:** Insert $x = u_t^{-\epsilon}$ (i.e. just below u_t) immediately before ℓ_t.
>
> Note that u_t lies above and precedes ℓ_{t-1} as u_t must be a 2 in a 21. Now suppose for the sake of contradiction that ρ' contains a copy of β (which, as noted previously, must use x). Since the maximum increasing sequence ending at x in ρ' has length t, then by Observation 1, we may assume that x occurs as the t^{th} element of this copy of β. But any further element of this copy of β is greater than $x = u_t^{-\epsilon}$, and hence greater than u_t in ρ. So ρ would contain a copy of β using an increasing sequence of length t ending at u_t instead. Hence ρ' avoids β by Observation 3.3.

Case 2 u_t lies above and after ℓ_t, which lies above and after u_{t-1}.

Extension: Insert x immediately before and above ℓ_t.

If ρ' had a β it would have to use x, but then ρ would have a β using ℓ_t instead. Hence ρ' avoids β by Observation 3.3.

Case 3 u_t lies above but after ℓ_t. In addition ℓ_t lies below and after u_{t-1}.

Extension: Insert $x = u_t^{-\epsilon}$ immediately before ℓ_t.

This follows by a symmetric argument to Case 1.

□

By reverse complement which sends $\mathcal{C}_{k,l}$ to $\mathcal{C}_{l,k}$, it follows that each of the i^{th} last upper element must also be above and to the left of the i^{th} last lower element for $1 \leqslant i \leqslant l-1$.

Now for a plus irreducible permutation ρ of $\mathcal{C}_{k,l}$, we define the following property for $1 \leqslant j \leqslant l$:

(Property B_j) there exist a $21 \oplus 1^j$ above and to the right of ℓ_{k-1} in ρ;

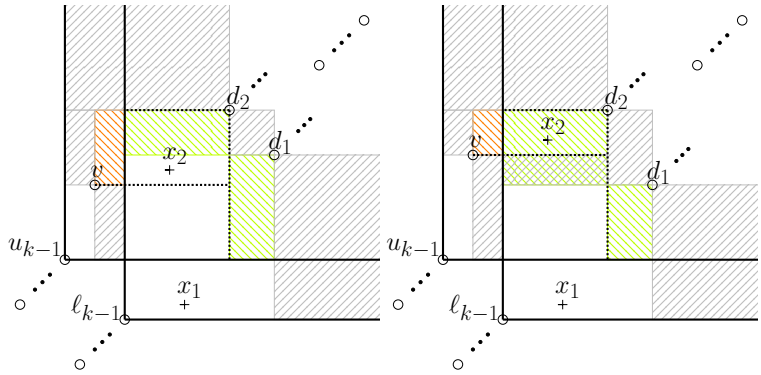
where ℓ_{k-1} is the last of its $k-1$ lower elements.

Proposition 3.11. *For any maximal plus irreducible ρ of $\mathcal{C}_{k,l}$, there exist a $21 \oplus 1^l$ above and to the right of the $(k-1)^{st}$ lower element ℓ_{k-1} in ρ.*

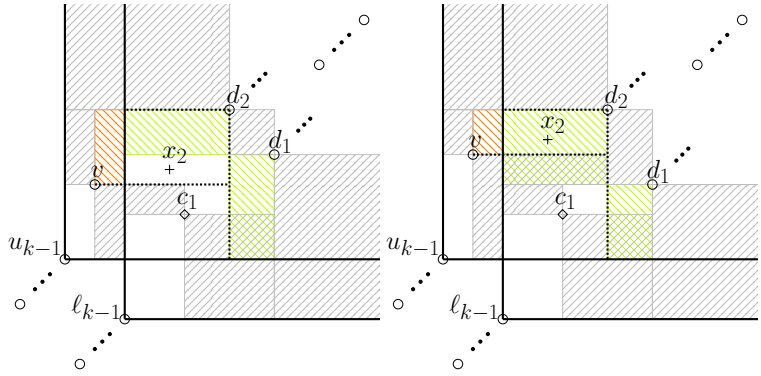
Proof. As in the previous argument, we prove by induction on j that, for each $1 \leqslant j \leqslant l$, if a plus irreducible permutation $\rho \in \mathcal{C}_{k,l}$ satisfies properties B_1 up to B_{j-1} but not B_j, then ρ is not maximal as we can extend it to a larger plus irreducible ρ' that satisfies B_j while still avoids $1^k \oplus 21 \oplus 1^l$. In this case however, we sometimes add two elements x_1 and x_2 to ρ in order to form ρ'. Otherwise the plan of attack is the same.

Let v be the rightmost upper element of ρ before ℓ_{k-1}. Suppose for $j = t \leqslant l$, Property B_j holds up to $t-1$ and no $21 \oplus 1^t$ exists above and after ℓ_{k-1} in ρ. Choose the lowest and leftmost copy of $21 \oplus 1^{t-1}$ lying above and to the right of ℓ_{k-1} (i.e., that copy which would be found by the greedy algorithm), and let d_2 and d_1 in ρ be the elements forming the 21 of that copy (so $d_2 d_1 \sim 21$). Note that d_2 is an upper element, so $d_2 > v$, but d_1 might be greater than or less than v – both cases are illustrated in Figure 3.2 where relevant. We extend ρ to ρ' by considering the following three cases:

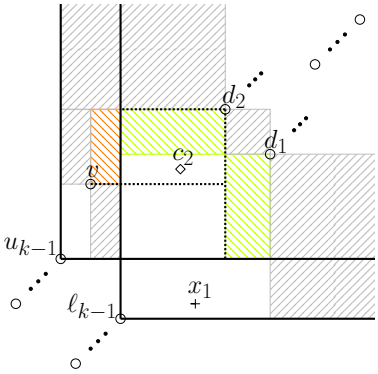
Case 1: No element exists after ℓ_{k-1} and before d_2 in ρ:



(a) Case 1: insert x_2 just above v, and x_1 right after x_2 and just above l_{k-1}



(b) Case 2: insert x_2 immediately before c_1 and above v



(c) Case 3: insert x_1 right after c_2 and just above l_{k-1}

Figure 3.2: Extending ρ to ρ' in Proposition 3.11. The shaded regions forbid elements of ρ due to 321 avoidance (gray), choice of d_2 and d_1 (lime), and choice of v (orange).

Extension: Create a 21 between ℓ_{k-1} and d_2 by adding $x_2 = v^{+\epsilon}$, and $x_1 = \ell_{k-1}^{+\epsilon}$ right after x_2.

Property B_t now holds in ρ', due to the created copy $x_2x_1 \oplus d_2 \oplus \ell^{t-1}$. Since x_2 and x_1 are the only elements in the rectangular region between ℓ_{k-1} and d_2 (by index and value), ρ' is clearly plus irreducible, as x_2 is separated from its predecessor and successor in value by ℓ_{k-1} and x_1 respectively, and x_1 is separated from its predecessor in value by x_2, and its successor in value precedes it by position.

It is easy to see that ρ' avoids 321. Now to show ρ' avoids $1^k \oplus 21 \oplus 1^l$, first note that by Lemma 3.10, u_{k-1} or ℓ_{k-1} is at most the $(k-1)^{\text{st}}$ element of any starting increasing sequence in ρ'. Thus we only need to show that extending to ρ' does not create any $1 \oplus 21 \oplus 1^l$ in the region above and after either u_{k-1} or ℓ_{k-1}. Suppose otherwise, since x_2 and x_1 are the first two elements above and after ℓ_{k-1} and form a 21, we could only use one of them, but then we would have a $21 \oplus 1^l$ above and after ℓ_{k-1} in ρ, a contradiction. For the u_{k-1} case we can only use x_2 and the same applies. Hence ρ' avoids β by Observation 3.3.

Case 2: There exists in ρ some element c_1 lying between ℓ_{k-1} and d_2 by position, and below v:

Extension: Choose c_1 to be the leftmost such element and add $x_2 = v^{+\epsilon}$ immediately before c_1.

Now ρ' satisfies B_t due to $x_2c_1 \oplus d_2 \oplus 1^{t-1}$. Here ρ' is plus irreducible since x_2's predecessor in value is v which is separated by ℓ_{k-1}, and its successor in value is separated by c_1.

In ρ, the rectangular region between ℓ_{k-1} and d_2 (by index and value) can contain no 21 as it would otherwise allow $21 \oplus 1^t$ above and after ℓ_{k-1}. Then it is easy to see that ρ' avoids 321. Now to show ρ' avoids $1^k \oplus 21 \oplus 1^l$, since x_2 and c_1 are the first two elements above and after ℓ_{k-1} and form a 21, if ρ' had a β it would have to use x_2 but not c_1, but then we would have a $21 \oplus 1$ above and after ℓ_{k-1} in ρ, contradiction. For the u_{k-1} case if ρ' had a β it would have to use x_2, but then ρ would have a β using v instead. Hence ρ' avoids β.

Case 3: No such c_1 described in Case 2 exists in ρ, but there is some element c_2 between ℓ_{k-1} and d_2 by position that lies above v and below d_1:

Extension: Choose c_2 to be the leftmost such element and add $x_1 = \ell_{k-1}^{+\epsilon}$ right

after c_2.

Note that this case does not exist if v lies below d_1, since by construction the leftmost d_2 was chosen, and thus no element can exist between ℓ_{k-1} and d_2 by position and between d_2 and d_1 (hence v) by value.

ρ' now satisfies B_t due to $c_2x_1 \oplus d_2 \oplus 1^{t-1}$. ρ' is plus irreducible since x_1's predecessor in value is ℓ_{k-1} which is separated by c_2, and x_1's successor in value precedes it by position. β avoidance in ρ' is also straightforward here as the region above and after u_{k-1} is unaffected by our extension.

□

In the structure covered by Lemma 3.10, the inverse map sends the first $k-1$ upper elements of ρ to the first $k-1$ lower elements. It thus follows by a symmetric argument to Proposition 3.11 that there must also exist a $21 \oplus 1^l$ above and to the right of u_{k-1} in ρ.

By reverse-complement which sends $\mathcal{C}_{k,l}$ to $\mathcal{C}_{l,k}$, we also have the symmetric case of Proposition 3.11 on the $(l-1)^{\text{st}}$ last upper and lower elements:

Corollary 3.12. *For any maximal plus irreducible ρ of $\mathcal{C}_{k,l}$, there exist a $1^k \oplus 21$ below and to the left of the $(l-1)^{st}$ last lower and upper element in ρ.*

Let $\rho \in \text{MPI}(\mathcal{C}_{k,l})$. Proposition 3.11 and Corollary 3.12 imply that if we substituted 12 for any of the first $k-1$ or last $l-1$ upper or lower elements of ρ then we would create a β. On the other hand, for any of the remaining elements, they are already preceded by 1^{k-1} and followed by 1^{l-1} so, given that they do not already form part of a β, they can be inflated by an increasing sequence of any length without creating a β. That is:

Theorem 3.13. *The set $\tilde{G}$ of* pMPI *of $\mathcal{C}_{k,l}$ consists of all elements $\tilde{\rho}$ derived from the maximal plus irreducibles $\rho \in \text{MPI}(\mathcal{C})$ using the following sign decorations: the first $k-1$ and the last $l-1$ upper and lower elements of each ρ are decorated with $\bullet$; and the rest of ρ are decorated with $+$.*

Corollary 3.14. *Every permutation in Grid($\tilde{G}$) is in the class $\mathcal{C}_{k,l}$.*

Proof. For any $\rho \in \text{MPI}(\mathcal{C})$, suppose that inflating ρ on elements other than the first $k-1$ and the last $l-1$ upper and lower elements by strictly increasing sequences creates an instance of $1^k \oplus 21 \oplus 1^l$. Then since the first $k-1$ upper and lower elements avoid 1^k, and the last $l-1$ upper and lower elements avoid 1^l, the inflation must have created

an increasing sequence that contains 1^k before any $21 \oplus 1^l$, and one that contains 1^l after $1^k \oplus 21$. Then the orginal (uninflated) permutation must contain $1^k \oplus 21 \oplus 1^l$ which is a contradiction. Hence any permutation inflated from ρ according to the sign decorations in Theorem 3.13 must be in Av(321, β). □

3.4.2 The algorithm

We have defined the set $\tilde{G}$ of peg permutations in the last section. As in the paper [HV15], the algorithm we describe outputs the set $\tilde{H}$ of peg permutations so that $\mathcal{C} = \mathrm{Grid}(\tilde{G}) = \biguplus_{\tilde{\rho} \in \tilde{H}} \tilde{\rho}[\mathcal{V}_{\tilde{\rho}}]$ as specified in Theorem 3.3. We do not, however, explicitly compute the convex set $\mathcal{V}_{\tilde{\rho}}$ associated with each $\tilde{\rho}$, instead we fix each $\mathcal{V}_{\tilde{\rho}}$ to be the set of all vectors that fills $\tilde{\rho}$.

Here we briefly outline the differences between our algorithm and the one in [HV15]. The Homberger-Vatter algorithm first constructs a complete set of peg permutations by taking the downset of $\tilde{G}$, then trims the set $\tilde{G}$ downwards to $\tilde{H}$ by removing duplicates through the compact-clean-combine process. Our algorithm instead only looks at the valid peg permutations and builds the set $\tilde{H}$ upwards - by first including the plus irreducible pegged subpermutations of each $\tilde{\rho} \in \tilde{G}$, then inflating these with appropriate sign restrictions, to obtain valid plus reducible pegged subpermutations not covered in the previous operation.

The reason for our approach is straightforward. Since the class $\mathcal{C}_{k,l}$ avoids 321, we have a much smaller set of reducible peg permutations to work with - here the only minus reducible interval possible in any peg permutations in $\mathcal{C}_{k,l}$ is $2^\bullet 1^\bullet$, while the only plus reducible intervals come in the form $1^\bullet 2^\bullet \dots r^\bullet$, $r \geqslant 2$. Only the latter need to be constructed explicitly by inflating the plus irreducibles with dotted elements, and by doing so, bypassing the compacting and cleaning steps, which can be very costly when the MPIs in the set $\tilde{G}$ are very large and many redundancies exist.

Our algorithm can be summarised into five steps, each of which will later be described in its own subsection.

1. Compute the set $\mathcal{E}$ of plus irreducible permutations of Av$(321, \beta)$. Then iterate through elements of $\mathcal{E}$ to find the set MPI($\mathcal{C}$).

2. For each $\rho \in$ MPI($\mathcal{C}$), compute its associated peg permutation $\tilde{\rho} \in \tilde{G}$ (using Theorem 3.13) and the set DR(ρ) of *dot-reduced* permutations.

3. For each permutation $q \in \mathcal{E}$, compute its set $\mathcal{P}_q$ of *maximal peggings* $\tilde{q}$ by checking its containment in the dot-reduced permutations of each $\rho \in \mathrm{MPI}(\mathcal{C})$.

4. Inflate the dotted elements of $\tilde{q}$ by increasing intervals of dotted elements, and keep the plus reducible peg permutations that avoid 321 and β.

5. Enumerate the peg permutations from the maximal peggings created from the above steps, and compute the generating function.

Before providing the details of our algorithm we need a few definitions. First note that in $\mathcal{C}_{k,l}$ the partial order on peg permutations remains the same as the general case previously described, except that we do not have the case where a permutation is decorated with a '$-$'. Under the peg permutation order, we call $\tilde{\tau}$ a *pegged subpermutation* of $\tilde{\rho}$ if $\tilde{\tau} \leqslant \tilde{\rho}$.

We then define the *maximal peggings* $\tilde{q}$ of $q \in \mathcal{E}$ to be the largest peg permutation by the aforementioned partial order, for the underlying permutation q. For example, the maximal pegging of $q = 213$ contained in $2^\bullet 1^\bullet 4^+ 3^+$ is $\tilde{q} = 2^\bullet 1^\bullet 3^+$. Here the other possible pegging of 213 is $2^\bullet 1^\bullet 3^\bullet$, which is smaller than $\tilde{q}$.

Definition 3.2. *For each* pMPI $\tilde{\rho} \in \tilde{G}$*, the set* $\mathrm{DR}(\rho)$ *of dot-reduced permutations consists of all pegged subpermutations of* $\tilde{\rho}$ *resulting from deleting some or all dotted elements of* $\tilde{\rho}$*.*

For example, the dot-reduced set of $\tilde{\rho} = 2^\bullet 1^\bullet 4^+ 3^+$ is $\mathrm{DR}(\rho) = \{2^+1^+, 1^\bullet 3^+ 2^+\}$. Now if some plus irreducible permutation $q \leqslant \rho$ is contained in 2143 and 132 but not 21, then the maximal pegging $\tilde{q}$ contained in $\tilde{\rho}$ must make use of the dotted element $1^\bullet$. Any other pegging of q also must have at least one dotted element.

This idea of containment checking is central to our algorithm and will later be used to find the maximal pegging(s) for each $q \in \mathcal{E}$ and its plus reducible extensions (steps 3 and 4).

Step 1. Computing $\mathcal{E}$ and $\mathrm{MPI}(\mathcal{C})$

Let $\mathcal{E}$ be the set of plus irreducible permutation in the class $\mathcal{C}_{k,l}$, and n be the maximum length of elements in $\mathcal{E}$. Albert, Atkinson, and Brignall proved in [AAB07] that $\mathcal{E}$ (and hence $\mathrm{MPI}(\mathcal{C})$) is finite, and provided the formula to calculate the length upper bound n. Below we state the result derived directly from Lemma 6 of [AAB07].

Corollary 3.15. *The length of any maximal plus irreducible* $\rho \in \mathrm{MPI}(\mathcal{C})$ *in the class* $\mathrm{Av}(321, 1^k \oplus 21 \oplus 1^l)$ *is at most*

$$n = \begin{cases} 4(k+l)-2 & \text{if } k > 0 \text{ and } l > 0 \\ 4(k+1) & \text{if } k = 0 \text{ or } l = 0 \end{cases}$$

The upper bound n given by Corollary 3.15 provides us with the terminating condition for computing $\mathcal{E}$. The plus irreducible set $\mathcal{E}$ can then be obtained by simply including every plus irreducible permutation in $\mathcal{C}$ for length from 1 to n. Note that $\mathcal{E}$ is closed downwards in the set of plus irreducible permutations ordered by containment.

The computation of $\mathrm{MPI}(\mathcal{C})$ now follows from Definition 3.1, by including all elements of $\mathcal{E}$ not contained in any other elements of $\mathcal{E}$. We initialise the set $\mathrm{MPI}(\mathcal{C})$ to be the elements of $\mathcal{E}$ of length n. Then we iterate over all permutations in $\mathcal{E}$ in order of decreasing length, and for each $q \in \mathcal{E}$, add q to $\mathrm{MPI}(\mathcal{C})$ if q is not contained in any existing $\rho \in \mathrm{MPI}(\mathcal{C})$.

The corresponding set $\tilde{G}$ of pMPI can then be computed from $\mathrm{MPI}(\mathcal{C})$ following Theorem 3.13. This set will be used to compute the set of dot-reduced permutations in the next section.

Step 2. Computing dot-reduced permutations for each $\tilde{\rho} \in \tilde{G}$

By Definition 3.2, the set $\mathrm{DR}(\rho)$ of dot-reduced permutations of each $\tilde{\rho} \in \tilde{G}$ can be computed by including all pegged subpermutations of $\tilde{\rho}$ that have fewer dots than $\tilde{\rho}$. Note that we do not include the original $\tilde{\rho}$ in $\mathrm{DR}(\rho)$.

The dot removal process can be easily done using a depth first traversal on the dotted elements of $\tilde{\rho} \in \tilde{G}$. For each dotted element c_i in $\tilde{\rho}$ we traversed to, remove c_i to obtain a pegged subpermutation $\tilde{\rho}'$, and add $\tilde{\rho}'$ to $\mathrm{DR}(\rho)$ if it is not already contained in $\mathrm{DR}(\rho)$. We then recursively perform the above steps on $\tilde{\rho}'$ to obtain a smaller subpermutation, and continue until the subpermutation we are working on contains no dot.

We detail these steps in the algorithm below:

Algorithm 5: Compute the dot-reduced set $\mathrm{DR}(\rho)$ for each $\tilde{\rho} \in \tilde{G}$

Input : A pegged maximal plus irreducible $\tilde{\rho} \in \tilde{G}$

Output: the corresponding set $\mathrm{DR}(\rho)$ of dot-reduced permutations

Initialisation: $\mathrm{DR}(\rho) := \{\tilde{\rho}\}$;

`removeDots(`$\tilde{\rho}$`)`: **if** $\tilde{\rho}$ *contains dotted elements* **then**

  foreach *dotted element* c_i *of* $\tilde{\rho}$ **do**

    $\tilde{\rho}' := \tilde{\rho}$ with c_i removed;

    if $\tilde{\rho} \notin \mathrm{DR}(\rho)$ **then**

      add $\tilde{\rho}'$ to $\mathrm{DR}(\rho)$

    `removeDots(`$\tilde{\rho}'$`)`

return $\mathrm{DR}(\rho)$

Step 3. Finding maximal peggings for each $q \in \mathcal{E}$

Let $q \in \mathcal{E}$ be a plus irreducible permutation in $\mathrm{Av}(321, \beta)$, we proceed to find its set of maximal peggings $\mathcal{P}_q$ as follows.

For each $\rho \in \mathrm{MPI}(\mathcal{C})$ containing q, we check whether q is contained in the underlying permutation g of each dot-reduced permutation $\tilde{g} \in \mathrm{DR}(\rho)$ computed in the last section. If $q \leqslant g$ for some $\tilde{g} \in \mathrm{DR}(\rho)$, let $f\colon q \to g$ be the embedding from q to g. For convenience, we say q is embedded to $\tilde{g}$ if there is an embedding $f\colon q \to g$ from q to the underlying permutation g of $\tilde{g}$, and $\tilde{g}$ contains q if $q \leqslant g$.

We want to make it so that each entry of q is embedded to the dotted elements of $\tilde{g}$ if and only if it is dotted. Note that this is not usually the case, as there may be multiple ways q can be embedded to g by mapping to different sets of dotted elements in $\tilde{g}$. For example, for $\tilde{g} = 1^\bullet 5^+ 2^\bullet 6^+ 3^+ 4^+ 7^+$, $q = 1324$ is involved in $g = 1526347$ by mapping q to either 1526 and 1637, the first of which suggests that both 1 and 2 in q can be dotted, while the second only allows the 1 in q to be dotted.

However, if $\tilde{g} \in \mathrm{DR}(\rho)$ is the smallest (by containment) dot-reduced permutation in $\mathrm{DR}(\rho)$ containing q, then the embedding from q to $\tilde{g}$ must involve all dotted elements of $\tilde{g}$. Moreover, for any dot-reduced permutation $\tilde{g}'$ containing $\tilde{g}$, the same elements must also be involved in the embedding from q to $\tilde{g}'$. Since we are looking for the maximal pegging(s) of q, it doesn't matter whether the additional dotted elements in $\tilde{g}'$ is also involved in the embedding from q to $\tilde{g}'$. Hence we only need to check whether

q is involved in the smallest dot-reduced permutations in $\mathrm{DR}(\rho)$. This can be done by iterating over the dot-reduced permutations in $\mathrm{DR}(\rho)$ in order of increasing length, and for each $\tilde{g} \in \mathrm{DR}(\rho)$ containing q, obtain a maximal pegging $\tilde{q}$ by decorating the elements of q embedded to dotted elements of $\tilde{g}$ as $\bullet$ and the rest with $+$, add $\tilde{q}$ to $\mathcal{P}_q$, then remove all $\tilde{g}' \in \mathrm{DR}(\rho)$ larger than $\tilde{g}$ from further consideration. The iteration continues until no further element of $\mathrm{DR}(\rho)$ can be considered.

Back to the above example, for $\tilde{g} = 1^\bullet 5^+ 2^\bullet 6^+ 3^+ 4^+ 7^+$, the smallest dot-reduced permutation here is $3^+4^+1^+2^+5^+$, which does not contain $q = 1324$. However, both 'one-dotted' permutations $1^\bullet 4^+ 5^+ 2^+ 3^+ 6^+$ and $4^+ 1^\bullet 5^+ 2^+ 3^+ 6^+$ contain 1324. And in either case the maximal pegging of q is $1^\bullet 3^+ 2^+ 4^+$.

We detail the steps of computing maximal pegging in the following algorithm. Here $\mathcal{D}_\rho$ is a copy of $\mathrm{DR}(\rho)$, and $\min[\mathrm{length}(\tilde{g})]$ denotes the length of the smallest dot-reduced permutation in $\mathcal{D}_\rho$.

Algorithm 6: Compute maximal pegging(s) $\tilde{q}$ for each $q \in \mathcal{E}$

Input : A plus irreducible permutation q in $\mathcal{C} = \mathrm{Av}(321, \beta)$

Output: the set $\mathcal{P}_q$ of maximal peggings $\tilde{q}$

Initialisation: $\mathcal{D}_\rho := \mathrm{DR}(\rho)$, $\mathcal{P}_q := \emptyset$;

foreach $\rho \in \mathrm{MPI}(\mathcal{C})$ *containing* q **do**
  for $i = \min[\mathit{length}(\tilde{g})]$ **to** $\mathit{length}(\tilde{\rho})$ **do**
    foreach $\tilde{g} \in \mathcal{D}_\rho$ *of length* i **do**
      if $q \leqslant g$ **then**
        compute (any) embedding $f \colon q \to g$;
        foreach *dotted element* $\tilde{g}_j$ *of* $\tilde{g}$ **do**
          decorate the element $f^{-1}(g_j)$ of q with a $\bullet$
        decorate the rest of the undecorated entries of q by $+$ to obtain $\tilde{q}$;
        if $\tilde{q} \notin \mathcal{P}_q$ **then**
          add $\tilde{q}$ to $\mathcal{P}_q$
        foreach $\tilde{g}' \in \mathcal{D}_\rho$ *with* $\tilde{g} < \tilde{g}'$ **do**
          remove $\tilde{g}'$ from $\mathcal{D}_\rho$

return $\mathcal{P}_q$

Step 4. Computing plus reducible permutations

We now proceed to compute the peggings of plus reducible permutations contained in the MPI of $\mathcal{C}_{k,l}$. This is done by inflating the pegged plus irreducibles (obtained in the last section) with increasing sequences of pegged elements.

Denote by $\mathcal{R}_{\tilde{q}}$ the set of plus reducibles in $\mathcal{C}_{k,l}$ obtained by inflating $\tilde{q}$. As in the Homberger and Vatter algorithm described previously, we want our resulting pegged plus reducibles to remain compact. Thus we only need to inflate the dotted elements of $\tilde{q}$ by $1^\bullet 2^\bullet$, for each $q \in \mathcal{E}$.

We inflate the dots using depth first traversal, in a way similar to the dot removal process in Step 2. For each dotted element $\tilde{c}_i \in \tilde{q} \in \mathcal{P}_q$, inflate $\tilde{c}_i$ by $1^\bullet 2^\bullet$ and add the resulting permutation $\tilde{q}'$ to $\mathcal{R}_{\tilde{q}}$ if it avoids β (321-avoidance follows automatically). We then recursively perform the above steps on $\tilde{q}'$ to obtain larger plus reducibles. Since there are $m = 2(k-1) + 2(l-1)$ dotted elements in an MPI in $\mathcal{C}_{k,l}$, the maximum number of dotted elements in an inflated permutation cannot exceed m. Thus the inflation process terminates when we have obtained all plus reducibles associated with $\tilde{q}$ with m dots.

Algorithm 7: Compute the set $\mathcal{R}_{\tilde{q}}$ of plus reducibles inflated from $\tilde{q}$

Input : A maximal pegging $\tilde{q} \in \mathcal{P}_q$ of q

Output: the set $\mathcal{R}_{\tilde{q}}$ of pegged plus reducibles inflated from $\tilde{q}$

Initialisation

$\mathcal{R}_{\tilde{q}} := \emptyset$;

$m := 2(k-1) + 2(l-1)$;

`inflateDots(`$\tilde{q}$`)`: **if** $\tilde{q}$ *contains fewer than* m *dotted elements* **then**

  foreach *dotted element* c_i *of* $\tilde{q}$ **do**

    $\tilde{q}' := \tilde{q}$ with c_i inflated by $1^\bullet 2^\bullet$;

    if $\tilde{q}'$ *avoids* β **then**

      add $\tilde{q}'$ to $\mathcal{R}_{\tilde{q}}$

    `inflateDots(`$\tilde{q}'$`)`

return $\mathcal{R}_{\tilde{q}}$

Step 5. Enumeration

The steps 1 to 4 above described the computation of the following set of peg permutations

- the set $\tilde{G}$ of pegged maximal plus irreducibles
- the set $\mathcal{P}_q$ of maximal peggings for every non-maximal plus irreducible $q \in \mathcal{E}$
- the set $\mathcal{R}_{\tilde{q}}$ of pegged plus reducibles inflated from each $\tilde{q} \in \mathcal{P}_q$, for any non-maximal $q \in \mathcal{E}$

We put together these steps below in Algorithm 8, in a single iteration through the elements $q \in \mathcal{E}$ in order of decreasing length. Now to compute the set $\tilde{H}$ from these components, note that we want every permutation in $\mathcal{C}_{k,l}$ to be obtainable from some element of $\tilde{H}$ by non-empty inflation. Thus we need to include in $\tilde{H}$ all peg permutations in $\tilde{G}$ as well as in all $\mathcal{P}$'s and $\mathcal{R}$'s.

Let $\mathcal{P}(\mathcal{C})$ be the union of $\mathcal{P}_q$ for all non-maximal $q \in \mathcal{E}$,

$$\mathcal{P}(\mathcal{C}) = \bigcup_{q \in \mathcal{E} \setminus \mathrm{MPI}(\mathcal{C})} \mathcal{P}_q \tag{3.1}$$

then $\mathcal{P}(\mathcal{C})$ is the set of all pegged non-maximal plus irreducible permutations.

Similarly we can obtain the set $\mathcal{R}(\mathcal{C})$ of all pegged plus reducible permutations, by taking the union of $\mathcal{R}_{\tilde{q}}$ for all $\tilde{q} \in \mathcal{P}(\mathcal{C})$:

$$\mathcal{R}(\mathcal{C}) = \bigcup_{\tilde{q} \in \mathcal{P}(\mathcal{C})} \mathcal{R}_{\tilde{q}} \tag{3.2}$$

Now the union of $\tilde{G}$, $\mathcal{P}(\mathcal{C})$ and $\mathcal{R}(\mathcal{C})$ is the set of maximal peggings of all compact (pegged) permutations in $\mathcal{C}_{k,l}$. Define the set of *down-peggings* of a maximal pegging $\tilde{\pi}$, denoted as $\mathrm{DP}(\tilde{\pi})$, to be the set of all peggings smaller than or equal to $\tilde{\pi}$ with the same underlying permutation π. For a set Π of maximal peggings, we define its down-pegging set $\mathrm{DP}(\Pi)$ to contain the down peggings of all elements in Π. Then the down-pegging set of $\tilde{G} \cup \mathcal{P}(\mathcal{C}) \cup \mathcal{R}(\mathcal{C})$ contains every compact peg permutation in $\mathcal{C}_{k,l}$. Hence we have

$$\tilde{H} = \mathrm{DP}(\tilde{G} \cup \mathcal{P}(\mathcal{C}) \cup \mathcal{R}(\mathcal{C})) \tag{3.3}$$

Algorithm 8: Computing the set $\tilde{H}$ of peg permutations

Input : A permutation $\beta = 1^k \oplus 21 \oplus 1^l$ with $k > l$

Output: The set $\tilde{H}$ of peg permutations

Initialisation

$\mathcal{E}$: set of plus irreducible permutations of Av(321, β);

n: maximum length of elements in $\mathcal{E}$;

for $q \in \mathcal{E}$, $|q| = n$ **do**

- add q to MPI($\mathcal{C}$);
- compute $\tilde{q}$ and add it to the set $\tilde{G}$ of pMPI (Theorem 3.13);
- compute dot-reduced set DR(q) of q (Algorithm 5)

for $i = n - 1$ **to** 1 **do**

- **foreach** $q \in \mathcal{E}$, $|q| = i$ **do**
  - **if** q *is not contained in any permutation in* MPI($\mathcal{C}$) **then**
    - add q to MPI($\mathcal{C}$);
    - compute $\tilde{q}$ and add it to the set $\tilde{G}$ of pMPI (Theorem 3.13);
    - compute dot-reduced set DR(q) of q (Algorithm 5)
  - **else**
    - compute the set $\mathcal{P}_q$ of maximal peggings of q (Algorithm 6);
    - **foreach** $\tilde{q} \in \mathcal{P}_q$ **do**
      - compute the set $\mathcal{R}_{\tilde{q}}$ of plus reducibles inflated from $\tilde{q}$ (Algorithm 7)

Compute $\mathcal{P}(\mathcal{C})$, $\mathcal{R}(\mathcal{C})$ and then $\tilde{H}$ by Equations (3.1) to (3.3);

return $\tilde{H}$

Coming to enumeration of the class $\mathcal{C}_{k,l}$. By Section 2.5 of Homberger and Vatter [HV15], the generating function of (the vector set associated with) a peg permutation consisting a single dotted element is x, and the generating function of a peg permutation consisting a single + element is $x^2/(1-x)$. Let

$$a = \frac{x^2}{1-x}$$

and

$$b = x.$$

Then a peg permutation in $\mathrm{Av}(321, \beta)$ of length i with j dotted elements has a generating function

$$a^{i-j}b^j$$

since it has j elements decorated with $\bullet$ and $i-j$ elements with $+$.

Now for any maximal pegging $\tilde{\pi}$ in the class $\mathcal{C}_{k,l}$, the enumeration is done by summing up the generating functions of elements in $\mathrm{DP}(\tilde{\pi})$. Let i be the length of $\tilde{\pi}$, and j be its number of dotted elements. First note that if $j = 0$, i.e. all elements of $\tilde{\pi}$ are decorated with $+$, then since any $+$ element of $\tilde{\pi}$ can be either a $\bullet$ or a $+$ in its down-peggings, the number of permutations in $\mathrm{DP}(\tilde{\pi})$ with k dotted elements and $i-k$ plus elements is $\binom{i}{k}$. Summing up thus gives the generating function $\sum \binom{i}{k} a^{i-k} b^k = (a+b)^i$. Now since any $\bullet$ element can only stay as a $\bullet$ in its down-pegging, the generating function of $\mathrm{DP}(\tilde{\pi})$ is given by

$$b^j (a+b)^{i-j}$$

for $0 \leqslant j \leqslant i$.

We can then enumerate the class $\mathcal{C}_{k,l}$ by adding the counts from the maximal peggings of $\tilde{G}$, $\mathcal{R}(\mathcal{C})$ and $\mathcal{P}(\mathcal{C})$, provided that $|\mathcal{P}_q| = 1$ for all $q \in \mathcal{E}$. Define a two dimensional array A, where each entry $A[i][j]$ stores the number of peg permutations of length i with j dotted elements. Then for each maximal pegging $\tilde{z}$ in $\tilde{G}$, $\mathcal{R}(\mathcal{C})$ or $\mathcal{P}(\mathcal{C})$ computed in Algorithm 8, we add the count of $A[|\tilde{z}|]$[number of dots in $\tilde{z}$] by 1. After summing all maximal peggings for the sets considered above, we get the generating function

$$C[A] = \sum_{\substack{0 \leqslant i \leqslant n \\ 0 \leqslant j \leqslant m}} A[i][j] b^j (a+b)^{i-j} \tag{3.4}$$

where $m = 2(k-1) + 2(l-1)$ is the maximum number of dots a peg permutation can have in $\mathcal{C}_{k,l}$.

Now for the more complicated case where $|\mathcal{P}_q| > 1$. Here we cannot enumerate the down-peggings by simply adding up counts in $A[i][j]$, as there are multiple maximal peggings associated with the same underlying permutation. Instead we compute the generating function individually for each $\mathcal{P}_q$.

Let $h_1, \ldots h_t$ be the indices the elements of which are decorated with $+$ in all $\tilde{q} \in \mathcal{P}_q$, and let t be the number of such indices. For example, in $\mathcal{P}_q = \{1^\bullet 3^+ 2^+ 4^+, 1^+ 3^+ 2^+ 4^\bullet\}$, the 3 and 2 are decorated with $+$ in both elements of $\mathcal{P}_q$, thus $h_1 = 2$, $h_2 = 3$ and $t = 2$. Now the intersection $\cap_{\tilde{q} \in \mathcal{P}_q} \mathrm{DP}(\tilde{q})$ of down-peggings of $\mathcal{P}_q$ consists of all permutations where each of their non-h_k^{th} element is dotted, and has a generating function

$$b^{i-t} (a+b)^t.$$

Let d_j denote the number of j dotted permutations contained in some $\tilde{q} \in \mathcal{P}_q$, where all its elements indexed h_k $(1 \leqslant k \leqslant t)$ are decorated $+$. Let s be the minimum number of dots of any $\tilde{q} \in \mathcal{P}_q$. Then the generating function for the non intersecting part is

$$\sum_{s \leqslant j < i-t} d_j a^{i-t-j} b^j (a+b)^t$$

The generating function of $\mathcal{P}_q$ is then

$$C[\mathcal{P}_q] = \sum_{s \leqslant j < i-t} d_j a^{i-t-j} b^j (a+b)^t + b^{i-t}(a+b)^t \tag{3.5}$$

Finally we sum up $C[A]$ and $C[\mathcal{P}_q]$ for all $|\mathcal{P}_q| > 1$ to obtain the generating function of the entire class $\mathcal{C}_{k,l}$

$$\mathrm{GF}(\mathcal{C}_{k,l}) = C[A] + \sum_{\{q \in \mathcal{E} : |\mathcal{P}_q| > 1\}} C[\mathcal{P}_q] \tag{3.6}$$

3.4.3 Examples

We now illustrate the above steps of our algorithm by showing a few examples. First we use selective examples to go through each step in the enumeration of the class $\mathcal{C}_{3,0} = \mathrm{Av}(321, 12354)$, which covers everything explained in Steps 1-4 of the algorithm. Here $\mathcal{C}_{3,0}$ is a class where $|\mathcal{P}_q| = 1$ for all $q \in \mathcal{E}$. i.e. every plus irreducible in $\mathcal{C}_{3,0}$ has only one maximal pegging. Thus in the final enumeration step (Step 5) we compute its generating function using Equation 3.4.

Step 1. Computing $\mathcal{E}$ and $\mathrm{MPI}(\mathcal{C})$

We first compute the list of plus irreducibles $\mathcal{E}$ of $\mathcal{C}_{3,0}$, which is the following:

$1,$

$21,$

$213, 132,$

$2413, 1324, 2143, 3142,$

$14253, 31524, 31425, 21354, 21435, 13254, 13524, 24135, 24153$

$\dots$

$4\,6\,8\,1\,9\,2\,10\,3\,5\,7\,11$

We then find the maximal plus irreducibles $\mathrm{MPI}(\mathcal{C}_{3,0})$ by iterating through elements of $\mathcal{E}$ in order of decreasing length. We found the the following four maximal plus irreducibles in $\mathcal{C}_{3,0}$:

$$\mathrm{MPI}(\mathcal{C}_{3,0}) = \{215738469, 4\,1\,6\,8\,2\,9\,3\,5\,7\,10, 2\,5\,7\,1\,8\,3\,9\,4\,6\,10, 4\,6\,8\,1\,9\,2\,10\,3\,5\,7\,11\}.$$

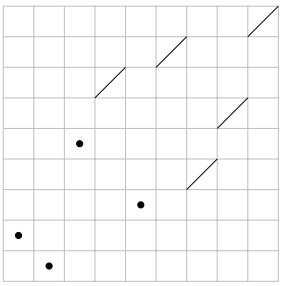
Using Theorem 3.13, we compute the set $\tilde{G}$ of pegged maximal plus irreducibles in the class, the elements of which are shown in Figure 3.3 below.

The class $\mathcal{C}_{3,0} = \mathrm{Av}(321, 12354)$ is thus the union of four grid classes

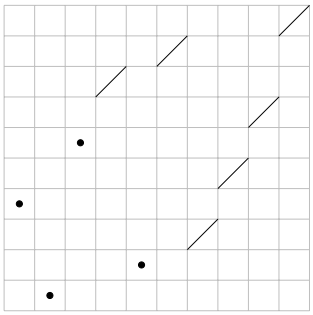
$$\begin{aligned}\mathrm{Av}(321, 12354) = &\ \mathrm{Grid}(2^\bullet 1^\bullet 5^\bullet 7^+ 3^\bullet 8^+ 4^+ 6^+ 9^+) \\ &\cup \mathrm{Grid}(4^\bullet 1^\bullet 6^\bullet 8^+ 2^\bullet 9^+ 3^+ 5^+ 7^+ 10^+) \\ &\cup \mathrm{Grid}(2^\bullet 5^\bullet 7^+ 1^\bullet 8^+ 3^\bullet 9^+ 4^+ 6^+ 10^+) \\ &\cup \mathrm{Grid}(4^\bullet 6^\bullet 8^+ 1^\bullet 9^+ 2^\bullet 10^+ 3^+ 5^+ 7^+ 11^+)\end{aligned}$$

Step 2. Computing dot-reduced permutations for each $\tilde{\rho} \in \tilde{G}$

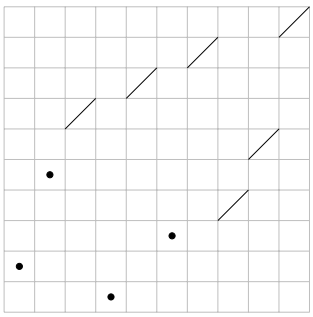
We then compute the dot-reduced set $\mathrm{DR}(\rho)$ associated with each $\tilde{\rho} \in \tilde{G}$ in the class. For example, for $\tilde{\rho}_2 = 4^\bullet 1^\bullet 6^\bullet 8^+ 2^\bullet 9^+ 3^+ 5^+ 7^+ 10^+$ shown in Figure 3.3(b), its dot-reduced set $\mathrm{DR}(\rho_2)$ consists of all permutations obtained by removing some or all dotted elements from $\tilde{\rho}_2$. Below we list all such permutations:



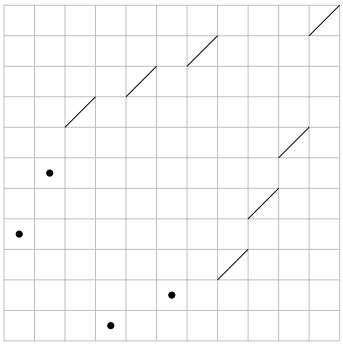
(a) $\tilde{\rho}_1 = 2^\bullet 1^\bullet 5^\bullet 7^+ 3^\bullet 8^+ 4^+ 6^+ 9^+$



(b) $\tilde{\rho}_2 = 4^\bullet 1^\bullet 6^\bullet 8^+ 2^\bullet 9^+ 3^+ 5^+ 7^+ 10^+$



(c) $\tilde{\rho}_3 = 2^\bullet 5^\bullet 7^+ 1^\bullet 8^+ 3^\bullet 9^+ 4^+ 6^+ 10^+$



(d) $\tilde{\rho}_4 = 4^\bullet 6^\bullet 8^+ 1^\bullet 9^+ 2^\bullet 10^+ 3^+ 5^+ 7^+ 11^+$

Figure 3.3: The permutations $\tilde{\rho} \in \tilde{G}$ for the class $\mathcal{C}_{3,0}$

3-dotted: $1^\bullet 5^\bullet 7^+ 2^\bullet 8^+ 3^+ 4^+ 6^+ 9^+, 3^\bullet 5^\bullet 7^+ 1^\bullet 8^+ 2^+ 4^+ 6^+ 9^+, 4^\bullet 1^\bullet 7^+ 2^\bullet 8^+ 3^+ 5^+ 6^+ 9^+,$
$3^\bullet 1^\bullet 5^\bullet 7^+ 8^+ 2^+ 4^+ 6^+ 9^+$

2-dotted: $4^\bullet 6^+ 1^\bullet 7^+ 2^+ 3^+ 5^+ 8^+, 1^\bullet 6^+ 2^\bullet 7^+ 3^+ 4^+ 5^+ 8^+, 1^\bullet 4^\bullet 6^+ 7^+ 2^+ 3^+ 5^+ 8^+,$
$3^\bullet 6^+ 1^\bullet 7^+ 2^+ 4^+ 5^+ 8^+, 2^\bullet 4^\bullet 6^+ 7^+ 1^+ 3^+ 5^+ 8^+, 3^\bullet 1^\bullet 6^+ 7^+ 2^+ 4^+ 5^+ 8^+$

1-dotted: $5^+ 1^\bullet 6^+ 2^+ 3^+ 4^+ 7^+, 3^\bullet 5^+ 6^+ 1^+ 2^+ 4^+ 7^+, 1^\bullet 5^+ 6^+ 2^+ 3^+ 4^+ 7^+, 2^\bullet 5^+ 6^+ 1^+ 3^+ 4^+ 7^+$

0-dotted: $4^+ 5^+ 1^+ 2^+ 3^+ 6^+$

The dot-reduced set for the other three pMPI are computed similarly.

Step 3. Finding maximal peggings for each $q \in \mathcal{E}$

We illustrate Step 3 and 4 of the algorithm on the example $q = 2143$. To find the maximal peggings of 2143, we first note that it is contained in all MPIs of $\mathcal{C}_{3,0}$, thus we need to perform q-containment checks on the dot-reduced elements in $\mathrm{DR}(\rho)$ for every $\tilde{\rho} \in \tilde{G}$.

We first check whether 2143 is contained in the 0-dotted element in each of the four dot-reduced sets. For example the 0-dotted element of $\mathrm{DR}(\rho_2)$ is $4^+ 5^+ 1^+ 2^+ 3^+ 6^+$, which does not contain 2143. Similarly, we can show that no 0-dotted element in any other dot-reduced set contains 2143.

It is routine to check that 2143 is also not contained in any 1-dotted element in any of the four dot-reduced sets. For the 2-dotted case, in $\mathrm{DR}(\rho_2)$ there are three permutations containing 2143: $4^\bullet 6^+ 1^\bullet 7^+ 2^+ 3^+ 5^+ 8^+$, $3^\bullet 6^+ 1^\bullet 7^+ 2^+ 4^+ 5^+ 8^+$, and $3^\bullet 1^\bullet 6^+ 7^+ 2^+ 4^+ 5^+ 8^+$, and in each case the 2 and 1 in 2143 are mapped to the dotted elements of the dot-reduced permutations. It is easy to show that the same happens for all other $\rho \in \mathrm{MPI}(\mathcal{C}_{3,0})$. Thus the set of maximal peggings for $q = 2143$ is $\mathcal{P}_{2143} = \{2^\bullet 1^\bullet 4^+ 3^+\}$.

Step 4. Computing plus reducible permutations

Continuing from the last step, after we found the maximal pegging $2^\bullet 1^\bullet 4^+ 3^+$, we inflate its dotted elements recursively by $1^\bullet 2^\bullet$ to compute its associating plus reducible permutations.

Inflating $2^\bullet$ in $2^\bullet 1^\bullet 4^+ 3^+$ gives $2^\bullet 3^\bullet 1^\bullet 5^+ 4^+$.

Inflating $1^\bullet$ in $2^\bullet 1^\bullet 4^+ 3^+$ gives $3^\bullet 1^\bullet 2^\bullet 5^+ 4^+$.

Both of these avoid 12354. We can thus inflate their dotted elements again to generate more plus reducible peg permutations. Now the only 12354-avoiding peg permutation obtainable from inflating the above permutations is $3^\bullet 4^\bullet 1^\bullet 2^\bullet 6^+ 5^+$ - by inflating either the $1^\bullet$ in $2^\bullet 3^\bullet 1^\bullet 5^+ 4^+$, or the $3^\bullet$ in $3^\bullet 1^\bullet 2^\bullet 5^+ 4^+$. Since the resulting peg permutation now has four dotted elements (the maximum number allowed), the inflation process terminates and we have $\mathcal{R}_{2^\bullet 1^\bullet 4^+ 3^+} = \{2^\bullet 3^\bullet 1^\bullet 5^+ 4^+, 3^\bullet 1^\bullet 2^\bullet 5^+ 4^+, 3^\bullet 4^\bullet 1^\bullet 2^\bullet 6^+ 5^+\}$.

Repeating Steps 3 and 4 on every $q \in \mathcal{E}$ and we obtain the complete sets of $\mathcal{P}_q$ and $\mathcal{R}_{\tilde{q}}$, in each case we take their union according to Equations (3.1) and (3.2) and obtain $\mathcal{P}(\mathcal{C}_{3,0})$ and $\mathcal{R}(\mathcal{C}_{3,0})$.

Step 5. Enumeration

Now the set $\tilde{H}$ can be computed from $\mathcal{P}(\mathcal{C}_{3,0})$, $\mathcal{R}(\mathcal{C}_{3,0})$ and $\tilde{G}$ using Equation (3.3). To obtain the generating function of the class, since for $\mathcal{C}_{3,0} = \mathrm{Av}(321, 12354)$, $|\mathcal{P}_q| = 1$ for every $q \in \mathcal{E}$, we can count the number of j-dotted peg permutations of length i in each $\mathcal{P}_q$, $\mathcal{R}_{\tilde{q}}$, and $\tilde{G}$ using the array $A[i][j]$, then compute the generating function of $\mathcal{C}_{3,0}$ using Equation (3.4).

For our example $q = 2143$, since we already have $\mathcal{P}_q = \{2^\bullet 1^\bullet 4^+ 3^+\}$ and $\mathcal{R}_{\tilde{q}} = \{2^\bullet 3^\bullet 1^\bullet 5^+ 4^+, 3^\bullet 1^\bullet 2^\bullet 5^+ 4^+, 3^\bullet 4^\bullet 1^\bullet 2^\bullet 6^+ 5^+\}$, the array counts are added as follows:

$$A[4][2] : +1$$
$$A[5][3] : +2$$
$$A[6][4] : +1$$

By Equation (3.4), the components involved then contribute to the generating function of the class $\mathcal{C}_{3,0}$ by adding the following terms:

$$b^2(a+b)^2 + b^3(a+b)^2 + b^4(a+b)^2.$$

Summing this with the terms for all other $q \in \mathcal{E}$ (the computations of which we omit due to space constraints) gives

$$\begin{aligned}\mathrm{GF}(\mathcal{C}_{3,0}) &= b^4(a+b)^7 + 7b^4(a+b)^6 + 19b^4(a+b)^5 + 2b^3(a+b)^6 + 25b^4(a+b)^4 \\ &+ 11b^3(a+b)^5 + 16b^4(a+b)^3 + 21b^3(a+b)^4 + 3b^2(a+b)^5 + 4b^4(a+b)^2 \\ &+ 16b^3(a+b)^3 + 9b^2(a+b)^4 + 4b^3(a+b)^2 + 8b^2(a+b)^3 + 2b(a+b)^4 + 2b^2(a+b)^2 \\ &+ 3b(a+b)^3 + b(a+b)^2 + (a+b)^3 + (a+b)^2 + (a+b) + 1\end{aligned}$$

The generating function of $\mathcal{C}_{3,0}$ is then obtained by substituting for $a = \frac{x^2}{1-x}$, $b = x$ and factoring the above sum:

$$\mathrm{GF}(\mathcal{C}_{3,0}) = \frac{x^9 - 4x^8 + 4x^7 - x^6 + 8x^5 - 21x^4 + 23x^3 - 16x^2 + 6x - 1}{(x-1)^7}$$

The case $|\mathcal{P}_q| > 1$

Now let's look at an example where we have multiple maximal peggings (i.e. $|\mathcal{P}_q| > 1$) for at least one $q \in \mathcal{E}$. The smallest class having this property is $\mathcal{C}_{2,2} = \mathrm{Av}(321, 1^2 \oplus 21 \oplus 1^2)$, which contains over 30 MPIs and involves lengthy computations in its enumeration. As such, we only consider the example $q = 1324$ and illustrate the steps involved in computing the generating function of $\mathcal{P}_q$.

It is routine to check that the pegging $1^+3^+2^+4^+$ is impossible since 1324 is not contained in the 0-dotted element of any dot-reduced set. However, it is possible to peg 1324 with one dot in two different ways. Take for example the MPI $\rho = 4\,6\,1\,7\,10\,12\,2\,13\,3\,5\,8\,14\,9\,11$, which has an associated pegging

$$\tilde{\rho} = 4^\bullet 6^+ 1^\bullet 7^+ 10^+ 12^+ 2^+ 13^+ 3^+ 5^+ 8^+ 14^\bullet 9^+ 11^\bullet.$$

It is easy to see that both $1^\bullet 3^+ 2^+ 4^+$ and $1^+ 3^+ 2^+ 4^\bullet$ can be embedded to $\tilde{\rho}$. After checking 1324-containment on all 1-dotted elements in every dot-reduced set we have

$$\mathcal{P}_{1324} = \{1^\bullet 3^+ 2^+ 4^+, 1^+ 3^+ 2^+ 4^\bullet\}.$$

Now we count the number of peg permutations contained in the elements of $\mathcal{P}_{1324}$. Note that we cannot simply add up the $A[i][j]$ values of $1^\bullet 3^+ 2^+ 4^+$ and $1^+ 3^+ 2^+ 4^\bullet$ since the peg permutations contained in both of them (e.g. $1^\bullet 3^\bullet 2^\bullet 4^\bullet$) will be counted twice. Instead, we first consider the maximal peg permutation contained in both elements: $1^\bullet 3^+ 2^+ 4^\bullet$. Its associated generating function is

$$b^2(a+b)^2.$$

We then consider the non-intersecting peg permutations, i.e. the ones contained only in one of $1^\bullet 3^+ 2^+ 4^+$ and $1^+ 3^+ 2^+ 4^\bullet$. These permutations must have their middle two elements decorated with + like every $\tilde{q} \in \mathcal{P}_q$, and must also have fewer dots than the maximal intersecting permutation $1^\bullet 3^+ 2^+ 4^\bullet$. Since the minimum number of dots in any $\tilde{q} \in \mathcal{P}_q$ is $s = 1$, then according to Step 5 of the algorithm, $s \leqslant j < 2$, therefore $j = 1$. i.e. all such permutations must have exactly 1 dotted element.

It is then straightforward that the only non-intersecting peg permutations are $1^\bullet 3^+ 2^+ 4^+$ and $1^+ 3^+ 2^+ 4^\bullet$ themselves. The generating function associated to each one of them is

$$ab(a+b)^2$$

since any peg permutation contained in one of these must have identical first and last element decorations as its parent, while the two middle elements of such permutations can be either + or •.

We finally obtain the generating function of $\mathcal{P}_{1324}$ by summing up the generating function of each component using Equation (3.5):

$$C[\mathcal{P}_{1324}] = b^2(a+b)^2 + 2ab(a+b)^2.$$

3.5 Concluding remarks

In the previous sections we examined the structure of 321-avoiding polynomial classes, and provided an algorithm which enumerates these classes which have the form $\mathrm{Av}(321, \beta)$, $\beta = 1^k \oplus 21 \oplus 1^l$. However, the complexity of the algorithm has not yet been analysed. On the implementation side, one possible direction of further study is to look for run-time improvements on the algorithm and try to compute the generating functions of $\mathrm{Av}(321, \beta)$ for longer β, like what we did in Chapter 2. However, it is possible that optimising on the algorithm itself may have limited effect on the number of classes we can enumerate, as we believe the bottleneck of enumerating a class is the generation of all its plus irreducibles, whose number increases drastically as the length of β increases. Hence, it may be useful to look for a more efficient way of generating the set of all plus irreducible in a class $\mathcal{C}$, to replace the naive approach of simply generating every permutation in $\mathcal{C}$ for each length that is plus irreducible.

On the structural side, Theorem 3.13 has provided us with a complete description on the MPI's peggings for any 321-avoiding polynomial class. As for the structure

of the MPI themselves, we now have partial knowledge on the relative placement of dotted elements thanks to Lemma 3.10. But how about the elements decorated with +? Can we find any structural patterns among them? In particular, it would be interesting to know whether there exist any correlation between certain plus-element patterns with subclasses of $\mathrm{Av}(321, \beta)$ where β is of a particular form. We believe the answer to the above questions may help us in the generation the plus irreducibles if something concrete can be formulated, and would certain benefit us as a whole in the understanding of the structures of 321-avoiding polynomial classes.

References

[AA05] M. H. Albert and M. D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Math.*, 300(1-3):1–15, 2005.

[AAB07] Michael H. Albert, Mike D. Atkinson, and Robert Brignall. Permutation classes of polynomial growth. *Annals of Combinatorics*, 11(3-4):249–264, 2007.

[AAB+11] Michael H. Albert, Mike D. Atkinson, Mathilde Bouvel, Nik Ruškuc, and Vincent Vatter. Geometric grid classes of permutations. *ArXiv e-prints*, pages 1–28, 2011.

[ABRV16] Michael H. Albert, Robert Brignall, Nik Ruškuc, and Vincent Vatter. Rationality for subclasses of 321-avoiding permutations. pages 1–34, 2016.

[Alb] Michael Albert. Permlab: Software for permutation patterns.

[ALLV15] Michael H. Albert, Marie-Louise Lackner, Martin Lackner, and Vincent Vatter. The Complexity of Pattern Matching for 321-Avoiding and Skew-Merged Permutations. pages 1–12, 2015.

[Atk99] Mike D. Atkinson. Restricted Permutations. 6(4):383–406, 1999.

[BBL98] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. *Inf. Process. Lett.*, 65(5):277–283, March 1998.

[CW99] Timothy Chow and Julian West. Forbidden subsequences and Chebyshev polynomials. *Discrete Mathematics*, 204(1-3):119–128, 1999.

[HV06] Sophie Huczynska and Vincent Vatter. Grid classes and the Fibonacci dichotomy for restricted permutations. *Electronic Journal of Combinatorics*, 13(1 R):1–14, 2006.

[HV15] Cheyne Homberger and Vincent Vatter. On the effective and automatic enumeration of polynomial permutation classes. *Journal of Symbolic Computation*, pages 1–16, 2015.

[JK16] Vít Jelínek and Jan Kynčl. Hardness of Permutation Pattern Matching. (1):1–27, 2016.

[KK03] Tomá Kaiser and Martin Klazar. On growth rates of closed permutation classes. *Electronic Journal of Combinatorics*, 9(2 R):1–20, 2003.

[MV00] Toufik Mansour and A Vainshtein. Layered restrictions and Chebyshev polynomials. *Ann Comb*, 5(3-4):8, 2000.

[SS85] Rodica Simion and Frank W. Schmidt. Restricted Permutations. *European Journal of Combinatorics*, 6(4):383–406, 1985.

[Wat07] Steve Waton. *On Permutation Classes Defined by Token Passing Networks, Gridding Matrices and Pictures: Three Flavours of Involvement.* PhD thesis, Univ. of St Andrews, 2007.

[Wes96] Julian West. Generating trees and forbidden subsequences. *Discrete Mathematics*, 157(13):363–374, 1996.

Appendix A

Conjectured generating functions

A.1 Generating functions of $\mathrm{Av}(321, \pi)$ for $|\pi| = 5$

π	**generating function of** $\mathrm{Av}(321, \pi)$	**sequence enumerating** $\mathrm{Av}(321, \pi)$
21453 21534 23154 31254	$\frac{3t^7-18t^6+54t^5-81t^4+69t^3-34t^2+9t-1}{(2t-1)^3(t-1)^4}$	$1, 2, 5, 14, 41, 120, 342, 940, 2491, 6388, 15928, 38792, 92645, 217680, 504522, 1155924, 2622447, \ldots$
24153 31524	$\frac{-(t-1)(2t-1)^2}{t^4-9t^3+12t^2-6t+1}$	$1, 2, 5, 14, 41, 121, 355, 1033, 2986, 8594, 24674, 70757, 202814, 581272, 1666003, 4775323, 13688536, \ldots$
23514 25134 31452 41253	$\frac{-(2t-1)^3}{t^5+3t^4-16t^3+17t^2-7t+1}$	$1, 2, 5, 14, 41, 122, 364, 1083, 3208, 9462, 27812, 81545, 238696, 698005, 2040025, 5960779, 17415475, \ldots$
23451 51234 24513 34152 35124 41523 34512 45123	$\frac{t^2-3t+1}{(3t-1)(t-1)}$	$1, 2, 5, 14, 41, 122, 365, 1094, 3281, 9842, 29525, 88574, 265721, 797162, 2391485, 7174454, 21523361, \ldots$
12354 21345	$\frac{t^9-4t^8+4t^7-t^6+8t^5-21t^4+23t^3-16t^2+6t-1}{(t-1)^7}$	$1, 2, 5, 14, 41, 113, 277, 607, 1212, 2245, 3913, 6488, 10319, 15845, 23609, 34273, 48634, \ldots$
12453 12534 23145 31245	$\frac{-3t^5+14t^4-19t^3+15t^2-6t+1}{(2t-1)(t-1)^5}$	$1, 2, 5, 14, 41, 116, 307, 760, 1779, 3986, 8641, 18282, 38005, 78024, 158791, 321236, 647247, \ldots$
12435 13245	$\frac{t^8-2t^7+13t^6-29t^5+44t^4-39t^3+22t^2-7t+1}{(t-1)^8}$	$1, 2, 5, 14, 41, 117, 312, 768, 1749, 3712, 7403, 13982, 25181, 43499, 72438, 116784, 182937, \ldots$

π	**generating function of** $\mathrm{Av}(321,\pi)$	**sequence enumerating** $\mathrm{Av}(321,\pi)$
13425 14235	$\frac{t^9-7t^8+29t^7-72t^6+117t^5-122t^4+83t^3-36t^2+9t-1}{(2t-1)(t-1)^8}$	$1, 2, 5, 14, 41, 118, 321, 816, 1946, 4396, 9509, 19898, 40643, 81650, 162325, 320868, 632796, \ldots$
21354	$\frac{-8t^4+15t^3-14t^2+6t-1}{(2t-1)^2(t-1)^3}$	$1, 2, 5, 14, 41, 118, 325, 854, 2153, 5246, 12437, 28846, 65737, 147686, 327941, 721190, 1573193, \ldots$
13452 15234 23415 41235 14523 34125	$\frac{-2t^5+10t^4-16t^3+14t^2-6t+1}{(t^2-3t+1)(t-1)^4}$	$1, 2, 5, 14, 41, 119, 336, 924, 2492, 6636, 17536, 46137, 121095, 317434, 831571, 2177734, 5702191, \ldots$
13254 21435	$\frac{-11t^7+41t^6-83t^5+101t^4-76t^3+35t^2-9t+1}{(2t-1)^2(t-1)^6}$	$1, 2, 5, 14, 41, 120, 342, 938, 2470, 6262, 15359, 36638, 85415, 195428, 440340, 979818, 2157836, \ldots$
13524 14253 24135 31425	$\frac{-5t^4+12t^3-13t^2+6t-1}{(2t-1)(t^2-3t+1)(t-1)^2}$	$1, 2, 5, 14, 41, 120, 345, 972, 2691, 7348, 19855, 53230, 141871, 376466, 995705, 2627018, 6918101, \ldots$

A.2 Generating functions of $\mathrm{Av}(321,\pi)$ for $|\pi| = 6$

π	**generating function of** $\mathrm{Av}(321,\pi)$	**sequence enumerating** $\mathrm{Av}(321,\pi)$
123465 213456	$\frac{4t^{13}-24t^{12}+54t^{11}-49t^{10}+3t^9+14t^8+16t^7-47t^6+72t^5-83t^4+61t^3-29t^2+8t-1}{(t-1)^9}$	$1, 2, 5, 14, 42, 131, 401, 1132, 2869, 6575, 13838, 27143, 50221, 88488, 149588, 244055, 386110, \ldots$
123546 132456	$\frac{t^{12}-3t^{11}-t^{10}+3t^9+19t^8-57t^7+119t^6-155t^5+144t^4-90t^3+37t^2-9t+1}{(t-1)^{10}}$	$1, 2, 5, 14, 42, 131, 407, 1209, 3361, 8697, 20998, 47558, 101663, 206343, 399842, 743345, 1331604, \ldots$
123564 123645 231456 312456	$\frac{2t^9-4t^8-5t^7+26t^6-49t^5+67t^4-55t^3+28t^2-8t+1}{(2t-1)(t-1)^7}$	$1, 2, 5, 14, 42, 131, 405, 1190, 3271, 8426, 20517, 47698, 106860, 232553, 494865, 1035218, 2138001, \ldots$
124356	$\frac{t^{11}+4t^{10}+9t^9+19t^8-55t^7+119t^6-155t^5+144t^4-90t^3+37t^2-9t+1}{(t-1)^{10}}$	$1, 2, 5, 14, 42, 131, 409, 1229, 3477, 9202, 22812, 53196, 117273, 245635, 491142, 941543, 1737438, \ldots$
124365 214356	unknown	
124536 125346 134256 142356	$\frac{-t^{10}+37t^9-132t^8+293t^7-429t^6+443t^5-324t^4+164t^3-55t^2+11t-1}{(2t-1)(t-1)^{10}}$	$1, 2, 5, 14, 42, 131, 409, 1232, 3505, 9358, 23501, 55856, 126584, 275631, 580889, 1192966, 2402303, \ldots$
124563 126345 234156 412356	$\frac{-3t^7+17t^6-37t^5+56t^4-50t^3+27t^2-8t+1}{(t^2-3t+1)(t-1)^6}$	$1, 2, 5, 14, 42, 131, 409, 1244, 3652, 10370, 28670, 77731, 207944, 551458, 1454516, 3823920, 10034093, \ldots$
124635 125364 241356 314256	$\frac{2t^8-14t^7+49t^6-105t^5+130t^4-96t^3+42t^2-10t+1}{(t^2-3t+1)(2t-1)^2(t-1)^4}$	$1, 2, 5, 14, 42, 131, 411, 1269, 3823, 11231, 32264, 90972, 252653, 693246, 1883973, 5080992, 13620466, \ldots$

π	**generating function of** $\mathrm{Av}(321,\pi)$	**sequence enumerating** $\mathrm{Av}(321,\pi)$
125634 341256	$\frac{t^8-3t^7+17t^6-37t^5+56t^4-50t^3+27t^2-8t+1}{(t^2-3t+1)(t-1)^6}$	$1, 2, 5, 14, 42, 131, 409, 1245, 3661, 10417,$ $28858, 78374, 209937, 557256, 1470709,$ $3867988, 10152106, \ldots$
132465 213546	$\frac{-41t^8+164t^7-340t^6+434t^5-356t^4+189t^3-63t^2+12t-1}{(2t-1)^4(t-1)^5}$	$1, 2, 5, 14, 42, 131, 412, 1274, 3820, 11059,$ $30946, 83982, 221902, 572987, 1450608,$ $3610506, 8854952, \ldots$
132546	unknown	
132564 132645 231546 312546	$\frac{5t^{10}-43t^9+201t^8-502t^7+774t^6-790t^5+545t^4-252t^3+75t^2-13t+1}{(2t-1)^4(t-1)^6}$	$1, 2, 5, 14, 42, 131, 414, 1298, 3980, 11848,$ $34170, 95567, 259838, 688841, 1785854,$ $4540152, 11345952, \ldots$
134265 142365 214536 215346	$\frac{3t^{10}-25t^9+111t^8-297t^7+490t^6-537t^5+399t^4-199t^3+64t^2-12t+1}{(2t-1)^3(t-1)^7}$	$1, 2, 5, 14, 42, 131, 413, 1285, 3883, 11308,$ $31686, 85598, 223736, 568145, 1407227,$ $3412091, 8124521, \ldots$
134526 152346 145236	unknown	
134562 162345 234516 512346 145623 156234 345126 451236	$\frac{t^7-8t^6+25t^5-45t^4+45t^3-26t^2+8t-1}{(3t-1)(t-1)^6}$	$1, 2, 5, 14, 42, 131, 413, 1294, 4007, 12272,$ $37277, 112622, 339152, 1019457, 3061373,$ $9188486, 27571645, \ldots$
134625 152364 251346 314526	$\frac{-(3t^2-3t+1)(2t^5-12t^4+21t^3-18t^2+7t-1)}{(t^5+3t^4-16t^3+17t^2-7t+1)(t-1)^4}$	$1, 2, 5, 14, 42, 131, 414, 1304, 4061, 12474,$ $37814, 113354, 336830, 994463, 2923055,$ $8567150, 25065978, \ldots$
135246 142536	$\frac{-23t^7+87t^6-172t^5+193t^4-129t^3+51t^2-11t+1}{(t^2-3t+1)(2t-1)^3(t-1)^3}$	$1, 2, 5, 14, 42, 131, 413, 1288, 3928, 11673,$ $33835, 95938, 267034, 732083, 1982801,$ $5319104, 14163360, \ldots$
135264 142635 241536 315246	$\frac{-8t^6+35t^5-66t^4+63t^3-33t^2+9t-1}{(2t-1)(t^4-9t^3+12t^2-6t+1)(t-1)^2}$	$1, 2, 5, 14, 42, 131, 414, 1302, 4038, 12319,$ $37009, 109768, 322346, 939658, 2724782,$ $7872320, 22687481, \ldots$
135624 152634 341526 351246 145263 146235 245136 415236	$\frac{2t^6-12t^5+32t^4-39t^3+25t^2-8t+1}{(3t-1)(2t-1)(t-1)^4}$	$1, 2, 5, 14, 42, 131, 414, 1305, 4077, 12612,$ $38683, 117864, 357388, 1079977, 3255844,$ $9799723, 29464007, \ldots$
136245 142563 235146 412536	$\frac{-2t^7-14t^6+63t^5-104t^4+88t^3-41t^2+10t-1}{(2t-1)(t^5+3t^4-16t^3+17t^2-7t+1)(t-1)^2}$	$1, 2, 5, 14, 42, 131, 415, 1315, 4133, 12839,$ $39397, 119532, 359188, 1070988, 3174239,$ $9365976, 27546660, \ldots$

π	**generating function of** $\mathrm{Av}(321,\pi)$	**sequence enumerating** $\mathrm{Av}(321,\pi)$
213465	$\frac{-12t^6+30t^5-52t^4+49t^3-27t^2+8t-1}{(2t-1)^2(t-1)^5}$	$1, 2, 5, 14, 42, 131, 408, 1229, 3531, 9664, 25311, 63840, 156032, 371585, 866218, 1984071, 4478889, \ldots$
213564 213645 231465 312465	$\frac{6t^7-34t^6+91t^5-124t^4+95t^3-42t^2+10t-1}{(t-1)^3(2t-1)^4}$	$1, 2, 5, 14, 42, 131, 411, 1266, 3784, 10941, 30641, 83364, 221078, 573319, 1458039, 3645286, 8978260, \ldots$
214365	unknown	
214563 216345 234165 412365	$\frac{t^{10}-22t^9+131t^8-368t^7+624t^6-687t^5+502t^4-242t^3+74t^2-13t+1}{(t^2-3t+1)(2t-1)^3(t-1)^5}$	$1, 2, 5, 14, 42, 131, 414, 1300, 4011, 12109, 35770, 103599, 294969, 827831, 2295597, 6302779, 17163183, \ldots$
214635 215364 241365 314265	$\frac{-11t^8+70t^7-192t^6+302t^5-289t^4+171t^3-61t^2+12t-1}{(t^2-3t+1)^2(2t-1)^2(t-1)^3}$	$1, 2, 5, 14, 42, 131, 415, 1313, 4108, 12655, 38351, 114436, 336766, 979172, 2817755, 8037596, 22756128, \ldots$
215634 341265	$\frac{-10t^8+60t^7-154t^6+235t^5-226t^4+138t^3-52t^2+11t-1}{(t^2-3t+1)(2t-1)^2(t-1)^5}$	$1, 2, 5, 14, 42, 131, 414, 1299, 3999, 12026, 35335, 101678, 287418, 800537, 2202942, 6003079, 16229901, \ldots$
231564 312645	$\frac{2t^8-38t^7+125t^6-215t^5+219t^4-137t^3+52t^2-11t+1}{(2t-1)^4(t-1)^4}$	$1, 2, 5, 14, 42, 131, 413, 1286, 3900, 11453, 32551, 89720, 240558, 629447, 1612225, 4053146, 10024784, \ldots$
231645 312564	$\frac{2t^{10}+2t^9-76t^8+287t^7-555t^6+653t^5-493t^4+241t^3-74t^2+13t-1}{(t-1)^4(2t-1)^5}$	$1, 2, 5, 14, 42, 131, 414, 1298, 3984, 11899, 34538, 97548, 268704, 723749, 1910938, 4957182, 12659536, \ldots$
234561 612345 245613 345162 461235 516234 246135 415263 345612 561234 351624 356124 451623 456123	$\frac{-(3t-1)(t-1)}{t^3-6t^2+5t-1}$	$1, 2, 5, 14, 42, 131, 417, 1341, 4334, 14041, 45542, 147798, 479779, 1557649, 5057369, 16420730, 53317085, \ldots$
234615 261345 314562 512364	unknown	
235164 241563 316245 412635	$\frac{-(2t-1)^4}{t^6+6t^5-35t^4+50t^3-31t^2+9t-1}$	$1, 2, 5, 14, 42, 131, 415, 1316, 4148, 12967, 40218, 123946, 380237, 1163066, 3551867, 10839772, 33078970, \ldots$

π	**generating function of** $\mathrm{Av}(321,\pi)$	**sequence enumerating** $\mathrm{Av}(321,\pi)$
235614 341562 361245 512634 256134 346125 415623 451263	$\frac{t^5+3t^4-16t^3+17t^2-7t+1}{(t-1)(t^4+12t^3-16t^2+7t-1)}$	$1, 2, 5, 14, 42, 131, 417, 1340, 4321, 13941,$ $44947, 144764, 465808, 1497736, 4813338,$ $15464049, 49673574, \ldots$
236145 412563	unknown	
241635 315264	$\frac{-(t-1)^2(2t-1)^3}{2t^6-22t^5+52t^4-57t^3+32t^2-9t+1}$	$1, 2, 5, 14, 42, 131, 415, 1315, 4136, 12883,$ $39768, 121896, 371847, 1131173, 3436743,$ $10438968, 31717929, \ldots$
245163 416235	$\frac{t^4-9t^3+12t^2-6t+1}{5t^4-17t^3+17t^2-7t+1}$	$1, 2, 5, 14, 42, 131, 416, 1329, 4248, 13560,$ $43217, 137570, 437581, 1391266, 4422590,$ $14057635, 44683032, \ldots$
251364 314625	unknown	
251634 315624 341625 351264	$\frac{-(t^2-3t+1)(-1+2t)^2}{t^5-12t^4+28t^3-23t^2+8t-1}$	$1, 2, 5, 14, 42, 131, 416, 1328, 4234, 13446,$ $42509, 133846, 420069, 1315228, 4111263,$ $12838149, 40065545, \ldots$