On Dots in Boxes

or

Permutation Pattern Classes and Regular Languages

Ruth Hoffmann

Doctor of Philosophy in Computer Science

April 2015

University of St Andrews

Abstract

This thesis investigates permutation pattern classes in a language theoretic context. Specifically we explored the regularity of sets of permutations under the rank encoding. We found that the subsets of plus- and minus-(in)decomposable permutations of a regular pattern class under the rank encoding are also regular languages under that encoding. Further we investigated the sets of permutations, which in their block-decomposition have the same simple permutation, and again we found that these sets of permutations are regular languages under the rank encoding. This natural progression from plus- and minus-decomposable to simple decomposable permutations led us further to the set of simple permutations under the rank encoding, which we have also shown to be regular under the rank encoding. This regular language enables us to find the set of simple permutations of any class, independent of whether the class is regular under the rank encoding.

Furthermore the regularity of the languages of some types of classes is discussed. Under the rank encoding we show that in general the skew-sum of classes, separable classes and wreath classes are not regular languages; but that the direct-sum of classes, and with some restrictions on the cardinality of the input classes the skew-sum and wreath sum of classes in fact are regular under this encoding.

Other encodings such as the insertion encoding and the geometric grid encoding are discussed and in the case of the geometric grid encoding alternative and constructive ways of retrieving the basis of a geometric grid class are suggested.

The aforementioned results of the rank encoding have been implemented, amongst other previously shown results, and tested. The program is available and accessible to everyone. We show that the implementation for finding the block-decomposition of a permutation has cubic time complexity with respect to the length of the permutation. The code for constructing the automaton that accepts the language of all plus-indecomposable permutations of a regular class under the rank encoding has quadratic time complexity with respect to the alphabet of the language. The procedure to find the automaton that accepts the language of minus-decomposable permutations has complexity $\mathcal{O}(k^5)$ and we show that the implementation of the automaton to find the language of simple permutations under the rank encoding has time complexity $\mathcal{O}(k^5 2^k)$, where k is the size of the alphabet. Further we show benchmark testing on previous important results involving the rank encoding on classes and their bases.

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Professor Steve Linton for his patience and understanding over the last three and a half years. For his words of wisdom and advice, when it seemed that I was lost in the woods, or just going too fast for my own good. Without his expert guidance this thesis would not have been possible. Further I would like to express my appreciation to everyone in the CIRCA group, the GAP community and the staff at both the School of Computer Science and Mathematics and Statistics for their support and advice when it was needed.

A special thank you goes to my previous and current office mates and the Taint for your insane attempts at keeping me sane.

Thanks to all the people in St Andrews and outside who stuck by me in high and low times, supplying gin and banter while enduring my baking, cooking and insistent calls to the pub. You know who you are and I love you.

Finally, I am eternally and unspeakably grateful to my brother Scott and my parents Maria and Christoph, for being there and supporting me throughout my life, you got me here and made me who I am.

Declarations

I, Ruth Hoffmann, hereby certify that this thesis, which is approximately 23000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in September, 2011 and as a candidate for the degree of Doctor of Philosophy in September 2011 the higher study for which this is a record was carried out in the University of St Andrews between 2011 and 2015.

Date: Signature of candidate:

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date: Signature of supervisor:

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

Access to printed copy and electronic publication of thesis through the University of St Andrews.

Date: Signature of candidate:

Date: Signature of supervisor:

Contents

List of Figures

List of Notation

$12[\alpha_1, \alpha_2]$	plus-decomposition
$21[\alpha_1, \alpha_2]$	minus-decomposition
δ	transition function
Γ	output alphabet
$\mathrm{Geom}(M)$	geomentric grid class of M
$\mathrm{Grid}(M)$	(monotone) grid class of M
Λ	standard figure
$\langle \mathcal{C} \rangle$	wreath closure of $\mathcal{C}$
$\mathcal{L}\|_1^n$	union of all possible shifts of $\mathcal{L}$ up to n
$\|w\|$	length of w
$\mathcal{A}[\mathcal{B}]$	wreath product of $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \wr \mathcal{B}$	wreath product of $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{C}$	permutation pattern class
$\mathcal{C} \ominus \mathcal{D}$	skew sum of pattern classes $\mathcal{C}$ and $\mathcal{D}$
$\mathcal{C} \oplus \mathcal{D}$	direct sum of pattern classes $\mathcal{C}$ and $\mathcal{D}$
$\mathcal{C}_k$	set of permutations corresponding to $E_k(\mathcal{C})$
$\mathcal{G}(k, A)$	gap automaton with $\Sigma = \{1, \ldots, k\}$ and final states A
$\mathcal{L}^{(n)}$	set of words of $\mathcal{L}$ from 1 up to length $n \in \mathbb{N}$
$\mathcal{NS}_k$	set of non-simple permutations with rank up to k in their rank encodings
$\mathcal{S}_k$	set of simple permutations with rank up to k in their rank encodings
$\mathcal{T}^t$	transposed transducer $\mathcal{T}$
$\mathscr{P}_l$	language of prefixes of rank encoded permutations, where $\sum gs'(w) = l$
$\mathscr{Q}_{i,j}$	language of prefixes of rank encoded permutations, where in $gs'(w) = \langle g_1, \ldots, g_n \rangle$ there is a $g_x = i$, $1 \leq x \leq n$, and $\sum_{h=1}^{x-1} g_h = j$
Ω_k	set of permutations, with rank at most k in their rank encoding
$\pi(i)$	ith element of π
π, σ	permutations
$\pi \ominus \sigma$	skew sum of permutations π and σ

$\pi \oplus \sigma$	direct sum of permutations π and σ
$\pi_i = \pi(x) \dots \pi(y)$	subsequence of $\pi = \sigma[\alpha_1, \dots, \alpha_n]$ corresponding to α_i
Σ	(input) alphabet
$\sigma[\alpha_1, \dots, \alpha_n]$	block-decomposition or inflation of a permutation
$\sigma[\mathcal{C}_1, \dots, \mathcal{C}_n]$	inflation of a permutation by classes
$\sigma \preceq \pi$	permutation contaiment/involvement
ε	empty word
A^C	complement set of A
$Av(B)$	notation for a pattern class avoiding the basis B
E	rank encoding
$E(\hat{\Omega}_k)$	sublanguage of $E(\Omega_k)$ containing the words of length > 1
$E(\pi)[x, \dots, y]$	subword of $E(\pi)$ corresponding to π_i
$E_{G(M)}$	geometric grid class encoding over M
E_{ICF}	context free insertion encoding, achieved through insertion boundedness
E_{IR}	regular insertion encoding, achieved through slot boundness
E_I	insertion encoding
$E_k(\mathcal{C})$	language of all rank encoded permutations with rank at most k
E_R	rank encoding
$gs'(w)$	gap sizes of $w = E(\pi)[1, \dots, y]$, $y \leq \lvert\pi\rvert$
$gs(A)$	gap sizes of A, $\langle g_1, \dots, g_n \rangle$
M	$0/\pm 1$ matrix
$M^{\times q}$	refinement of M
$Si(\mathcal{C})$	set of all simple permutations of $\mathcal{C}$
$w = w_1 \dots w_n$	word
w_i	letter
$x + \mathcal{L}$	language of $\mathcal{L}$ shifted up by $x \in \mathbb{N}$

Chapter 1

Permutation Pattern Classes

The idea of permutation pattern classes has arisen from an exercise question in Knuth's Art of Computer Programming Volume 1 (section 2.2.1 exercise 5) [Knu97], where Knuth was looking into what turned to be now known as stack sorting of permutations. Later Robert Tarjan expanded the exercise into the research of sequences being sorted by different types of data structures. In [Tar72] he describes the data structures as graphs. Research into seeing which permutations or unsorted sequences can be sorted by different graphs or data structures was further pursued because of that [AMR02, B0́3, SV09]. A similar approach is to look at the permutations that are being generated by a graph or network when a ordered sequence is given [ALT97, ALR04, Wat07]. This concept gave rise to a natural encoding of the sets of permutations [ALT97, AAR03, ALR04, Wat07], which turned the sets of permutations over an infinite alphabet into regular languages.

Simion and Schmidt [SS85] were amongst the first to characterise and enumerate closed sets of permutations based on the patterns or permutations that they avoid. Further enumeration results were of interest as the original set of permutations as determined in Knuth's exercise, presented to be the Catalan numbers. Amongst other constructions the enumeration of the classes of data structures was investigated [Wes95, B0́3, Atk99]. These enumeration results and the research into it were especially driven by a conjecture proposed by Herbert Wilf at the 1992 SIAM meeting, which stated that every permutation pattern class which avoids one permutation pattern has an exponential growth rate. This conjecture has been found to be true and been proven by Marcus and Tardos in [MT04].

Amongst these enumeration results the research extended into the languages of permutation sets and their generation through token passing networks, especially when a different proof to Knuth's exercise was shown in [ALT97]. At the same time an interest in being able to compute pattern classes and finding patterns in permutations developed [BBL98, UY00, AAAH01, XHP05]. An explicit language theoretic approach was first introduced in [AAR03] which also showed that it is possible to find the regular basis of a regular class and vice versa.

In [AA05] Albert and Atkinson stipulated that the knowledge of the set of simple permutations in a class is vital to the understanding of the whole class. This is based on the more general research of Schmerl and Trotter into binary relational structures [ST93], which permutations are a special case of. In fact, simple permutations are indeed crucial building blocks when it comes to pattern classes. As seen in [AAK03, BRV08, BHV08a, BHV08b, BRBP10, BRV10, AAB11a, AAB$^+$11b, Vat11, PR12, Bri12, ARV12, ASV12, AV13], simple permutations are vital for enumerating specific classes, building new permutations or classes, decomposing permutations, and even introducing a new type of classes. This new type of classes is grid classes, where we are looking at permutation classes consisting of permutations that have blocks of sequences order isomorphic to subpermutations placed in different positions [Wat07, VW11a, VW11b, Bri12, AV13, Bev13].

In this thesis we will be discussing different types of encodings of permutations and the language theoretic consequences of these encodings. Especially we will be looking at the rank encoding, which is the natural encoding of permutations generated by token passing networks. We will show the regularity of sets of plus- and minus-(in)decomposable permutations and σ-decomposable permutations. Further we will prove that the set of simple permutations under the rank encoding is regular and that we can find the set of all simple permutations in a non-regular class. Additionally we discuss the insertion encoding and the encoding of geometric grid classes, where for the latter we suggest constructive ways of finding the basis of a geometric grid class. Finally, for the results involving the rank encoding, we have written implementations. This program is available for anyone to use.

1.1 The Very Basics

First, here are several basic but important definitions before we start talking about permutation pattern classes.

Definition 1. A *word* $w = w_1 \dots w_n$ is a sequence of symbols w_i (called *letters*) which lie in an ordered set called the *alphabet*. An alphabet can be a finite or infinite set. A word can be empty. The *length* of a word $w = w_1 \dots w_n$ is denoted as $|w| = n$, where n is the number of letters of w. The empty word has length 0.

Definition 2. A *subsequence* of a word w is a word itself and is obtained by removing some or no letters of w and preserving the order of the remainder.

Definition 3. A *factor* of a word w is a consecutive subsequence of w.

Definition 4. A *permutation* π is a bijective function of a set onto itself.

In permutations, we will be typically using the set $[n] = \{1, \dots, n\}$, where $n \in \mathbb{N}$ and allow for the possibility that the set can also be empty. A permutation can be represented in *two-line notation* $\pi = \left(\begin{smallmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{smallmatrix}\right)$; in *cycle notation* as a sequence of cycles $(x, \pi(x), \pi(\pi(x)), \dots)$ for $x \in [n]$, until the image reaches x; as a *sequence* $\pi(1)\pi(2)\dots\pi(n)$ of the set $\{1, \dots, n\}$, which is the same as reading the lower line of the two-line notation; or as a *plot* where $(i, \pi(i))$ are points of the permutation drawn on a discrete (x, y) plane.

A permutation $\pi = \pi(1) \dots \pi(n)$ can be interpreted as a word in which the letters are distinct and taken from the alphabet $\{1, \dots, n\}$, $n \in \mathbb{N}$. The sequence or word notation of a permutation is the main notation that will be used.

Example. *The permutation* $\pi = 465312$ *is in cycle notation* $\pi = (1435)(26)$*, in two-line notation* $\pi = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 6 & 5 & 3 & 1 & 2 \end{smallmatrix}\right)$*, and figure 1.1 shows the plot of* π.

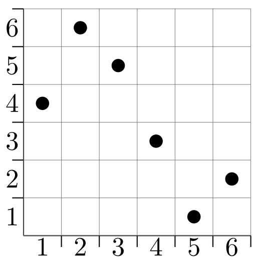


Figure 1.1: Plot of $\pi = 465312$.

1.2 General Permutation Pattern Classes

Definitions. *order isomorphism, involvement, containment*

Two sequences $\pi = \pi(1), \ldots, \pi(n)$ and $\sigma = \sigma(1), \ldots, \sigma(n)$ of the same length are said to be *order isomorphic* if, for all i, j, $\pi(i) \leq \pi(j)$ if and only if $\sigma(i) \leq \sigma(j)$ [Atk99]. The notion of patterns in permutations is known as *involvement* or *containment*. We say that a permutation $\sigma = \sigma(1) \ldots \sigma(n)$ is *contained* or *involved* in a permutation $\pi = \pi(1) \ldots \pi(m)$, where $n \leq m$, if there is a subsequence in π that is order isomorphic to σ. This is denoted by $\sigma \preceq \pi$. The containment order is a partial order on the set of all permutations [Bri10].

Example. 231 $\preceq$ 465312 *as* 231 *is order isomorphic to* 453*, along with other subsequences.*

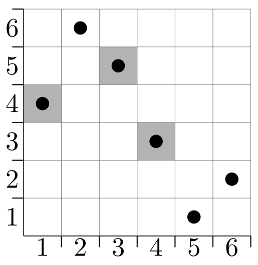


Figure 1.2: Plot of 465312 with an occurrence of 231 indicated.

Definitions. *permutation pattern class, basis*

A *permutation pattern class* is a set of permutations closed downwards under the containment order. In other words, a set $\mathcal{C}$ of permutations is a pattern class if $\pi \in \mathcal{C}$ and $\sigma \preceq \pi$ implies $\sigma \in \mathcal{C}$. Unless otherwise mentioned, if we say a set is closed, we mean that it is closed downwards under the above mentioned order. The complement set, $\mathcal{C}^C$, of a pattern class $\mathcal{C}$ is closed upwards under the containment order, i.e. if $\pi \in \mathcal{C}^C$ with $\pi \preceq \sigma$ then $\sigma \in \mathcal{C}^C$. The set of minimal permutations of $\mathcal{C}^C$ is called the *basis* B of $\mathcal{C}$ and we can describe pattern classes

$$\mathcal{C} = Av(B) = \{\pi : \sigma \not\preceq \pi, \text{ for all } \sigma \in B\}$$

as the set of permutations avoiding the basis [AAR03]. The class $\mathcal{C}$ can be represented by its basis as $Av(B)$.

Example. *The class of all strictly increasing permutations,* $\mathcal{C} = \{1, 12, 123, 1234, 12345, \ldots\}$ *has basis* $B = \{21\}$*. So* $\mathcal{C}$ *is* $Av(21)$*.*

In [Atk99] M. D. Atkinson states the following theorems on the construction of pattern classes based on the knowledge of other pattern classes.

Theorem 5. *[Atk99] Suppose that $\mathcal{C}$ and $\mathcal{D}$ are closed sets. Then $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$ are also closed. Moreover, if $\mathcal{C}$ and $\mathcal{D}$ each have a finite basis then both $\mathcal{C} \cap \mathcal{D}$ and $\mathcal{C} \cup \mathcal{D}$ have a finite basis.*

Theorem 6. *[Atk99] Suppose that $\mathcal{C}$ and $\mathcal{D}$ are closed. Let $[\mathcal{C}, \mathcal{D}]$ be the set of all permutations which are concatenations $\sigma\pi$, where σ is order isomorphic to a permutation in $\mathcal{C}$ and π is order isomorphic to a permutation in $\mathcal{D}$. Then $[\mathcal{C}, \mathcal{D}]$ is closed. Moreover, if $\mathcal{C}$ and $\mathcal{D}$ are each finitely based then so is $[\mathcal{C}, \mathcal{D}]$.*

These theorems allow us to construct new classes from the basic ones that are known. Further interest lies in the different types of classes, which are bound to permutations with the same properties.

1.3 Separable Classes

Definitions. *direct sum of permutations, skew sum of permutations*

The research into constructing pattern classes from smaller or known pattern classes has been extensive. Particularly, research in observing the enumerative properties of the constructed classes [Wes95]. Separable classes come from the idea of the direct and skew sum of permutations, which are the generalisation of two special cases of block-decomposition of permutations (see section 3.1).

The *direct sum of permutations* π and σ, with lengths m, n respectively, is defined as

$$(\pi \oplus \sigma)(i) = \begin{cases} \pi(i) & \text{if } 1 \leq i \leq m \\ \sigma(i-m) + m & \text{if } m+1 \leq i \leq m+n \end{cases}$$

[AAV11].

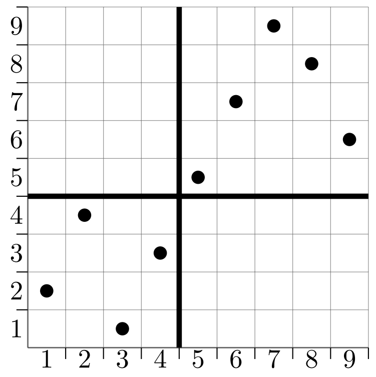
The *skew sum of permutations* π and σ, with lengths m, n respectively, is defined as

$$(\pi \ominus \sigma)(i) = \begin{cases} \pi(i) + n & \text{if } 1 \leq i \leq m \\ \sigma(i-m) & \text{if } m+1 \leq i \leq m+n \end{cases}$$

[AAV11].

Example. *Let $\pi = 2413$ and $\sigma = 13542$ then*

$$\pi \oplus \sigma = 241357986$$
$$\pi \ominus \sigma = 796813542.$$



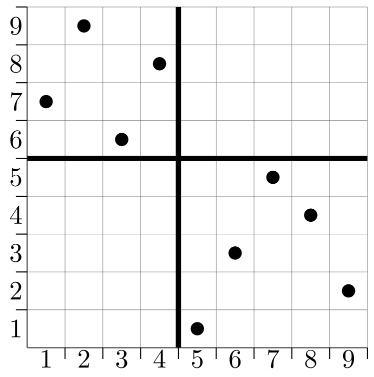


Figure 1.3: Plots of the direct and skew sums of 2413 and 13542.

Definitions. *direct sum of pattern classes, skew sum of pattern classes, separable pattern classes*

Following the definitions of the direct and skew sums of permutations, the direct and skew sums of pattern classes are defined as follows. The *direct sum of pattern classes* $\mathcal{C}, \mathcal{D}$ is

$$\mathcal{C} \oplus \mathcal{D} = \{\rho : \rho = \pi \oplus \sigma,\ \pi \in \mathcal{C},\ \sigma \in \mathcal{D}\}.$$

The *skew sum of pattern classes* $\mathcal{C}, \mathcal{D}$ is

$$\mathcal{C} \ominus \mathcal{D} = \{\rho : \rho = \pi \ominus \sigma,\ \pi \in \mathcal{C},\ \sigma \in \mathcal{D}\}.$$

Combining the notions of direct sums and skew sums leads us to another pattern class.

Proposition 7. *[AAV11] The class of separable permutations is the smallest non-empty class $\mathcal{C}$ that satisfies both $\mathcal{C} \oplus \mathcal{C} \subseteq \mathcal{C}$ and $\mathcal{C} \ominus \mathcal{C} \subseteq \mathcal{C}$.*

The basis of the class containing all separable permutations is $\{2413, 3142\}$ [BBL98].

1.4 Wreath Closed Classes

Definitions. *interval, block*

To define wreath classes we have to start at the definition of the wreath product (or inflation) of permutations. An *interval* (or *block* see [AA05]) in a permutation σ is a factor of contiguous values of σ.

Example. *In $\pi = 346978215$, $\pi(4)\pi(5)\pi(6) = 978$ is an interval and a factor, whereas $\pi(1)\pi(2)\pi(3)\pi(4) = 3469$ is just a factor.*

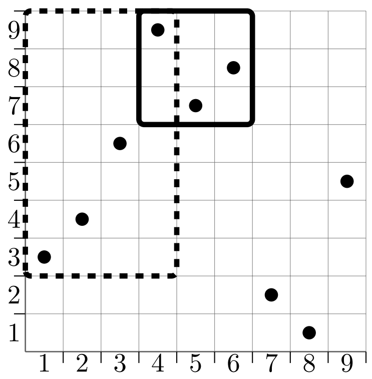


Figure 1.4: $\pi = 346978215$ with an interval (solid square) and a factor (dashed rectangle) indicated.

Definitions. *inflation, block-decomposition, deflation of permutations*

Given a permutation $\sigma = \sigma(1)\dots\sigma(m)$ of length m and non-empty permutations $\alpha_1, \dots, \alpha_m$ the *inflation* of σ by $\alpha_1, \dots, \alpha_m$, written as $\sigma[\alpha_1, \dots, \alpha_m]$, is the unique permutation obtained by replacing each entry $\sigma(i)$ by an interval that is order isomorphic to α_i, where the relative ordering of the intervals corresponds to the ordering of the entries of σ [AA05]. Conversely, a *block-decomposition* or *deflation* [AA05] of a permutation π is any expression of π written as an inflation $\pi = \sigma[\alpha_1, \dots, \alpha_m]$.

Example. *The inflation of* $\sigma = 24513$ *with* $\alpha_1 = 12$, $\alpha_2 = 1$, $\alpha_3 = 312$, $\alpha_4 = 21$, $\alpha_5 = 1$ *is* $24513[12, 1, 312, 21, 1] = 346978215$. *In other words a possible block-decomposition of* 346978215 *is* $24513[12, 1, 312, 21, 1]$.

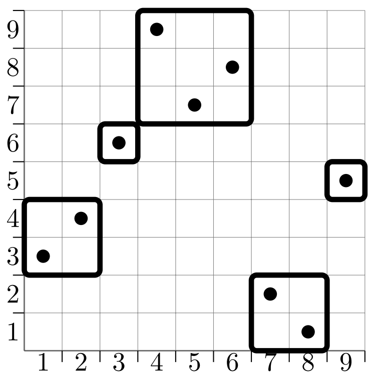


Figure 1.5: Plot of inflation $24513[12, 1, 312, 21, 1] = 346978215$.

Definitions. *inflation of classes, wreath product, wreath closed, wreath closure*

We can extend the definition of inflation of permutations to classes as follows

$$\sigma[\mathcal{C}_1, \dots, \mathcal{C}_n] = \{\sigma[\alpha_1, \dots, \alpha_n] : \alpha_i \in \mathcal{C}_i\}$$

where $|\sigma| = n$ [AA05] . This is a set of permutations, which are bound by the permutation σ in their decomposition. Furthermore, we can say that the *wreath product* [AS02, Bri07, Kit11] of two classes $\mathcal{A}$ and $\mathcal{B}$ is

$$\mathcal{A} \wr \mathcal{B} = \{\alpha[\beta_1, \dots, \beta_n] : \alpha \in \mathcal{A},\ \beta_1, \dots, \beta_n \in \mathcal{B}\}.$$

The wreath product is also sometimes denoted as $\mathcal{A}[\mathcal{B}]$. The following lemma is from [AS02], it shows that we can build further permutation classes through the wreath product.

Lemma 8 ([AS02])**.** *If $\mathcal{A}$ and $\mathcal{B}$ are closed then $\mathcal{A} \wr \mathcal{B}$ is closed.*

Now, we can define a class $\mathcal{A}$ to be *wreath closed* if $\mathcal{A} = \mathcal{A} \wr \mathcal{A}$, and in [AA05] the following proposition was shown.

Proposition 9 ([AA05])**.** *A class is wreath closed if and only if its basis consists entirely of simple permutations.*

See chapter 3 for more information on simple permutations and the block-decomposition of permutations.

The *wreath closure* $\langle \mathcal{A} \rangle$ of a pattern class $\mathcal{A}$ is the smallest wreath closed set that contains $\mathcal{A}$. We can describe $\langle \mathcal{A} \rangle$ as union of wreath products

$$\langle \mathcal{A} \rangle = \bigcup_{n=1}^{\infty} \mathcal{A}_n,$$

where $\mathcal{A} = \mathcal{A}_1$ and $\mathcal{A}_{n+1} = \mathcal{A} \wr \mathcal{A}_n$ [AA05].

Corollary 10 ([AA05])**.** *Let $\mathcal{A}$ be a wreath closed class. Then*

$$\mathcal{A} = \langle Si(\mathcal{A}) \rangle,$$

where $Si(\mathcal{A}) = \{\pi : \pi \in \mathcal{A},\ \pi \text{ simple}\}$.

1.5 Grid Classes

We now want to introduce a different way of representing classes. Similarly to the wreath product we are thinking of blocks in the plot of a permutation and how we position points in those blocks. Here we are given a grid on the plot of a permutation and we define the positioning of the points of the permutation in those cells. The behaviour of the permutations over the grid is described by matrices. Grid classes unify different theories concerning pattern classes of permutations of a specific form, amongst others. For example, the skew-merged permutations [HV06, AV13], which in the past were described as the permutations of a union of a decreasing subpermutation with an increasing subpermutation. Now this pattern class can be more easily defined as the grid class over the matrix $\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$ [HV06].

Remark 11. Please note that all matrices, in a grid class context, are indexed starting at the bottom left corner and the order is swapped. Namely, the entry ij of a matrix is in the i-th column from the left and j-th row from the bottom. This change in notation is due to the more natural correspondence to the plot representation of permutations.

Example. *The 3×4-matrix M is indexed in the following way*

$$\begin{pmatrix} (1,4) & (2,4) & (3,4) \\ (1,3) & (2,3) & (3,3) \\ (1,2) & (2,2) & (3,2) \\ (1,1) & (2,1) & (3,1) \end{pmatrix}$$

so $M_{2,3} = (2, 3)$ *denotes the entry in the second column from the left, and the third row from below.*

In general, the entries of matrices of grid classes are permutation classes, usually represented by their bases. We will be concentrating on matrices with $Av(21)$ or $Av(12)$ as entries. These grid classes are called *monotone grid classes.*

Definitions. *gridding matrix,* $0/\pm 1$ *matrix,* M*-gridding*

Let M be an $m \times n$ matrix, where the entries are either 0, 1 or -1. This matrix is called a *gridding matrix* [Wat07] or a $0/\pm 1$ *matrix* [AAB$^+$11b, VW11a]. An M*-gridding* of a permutation π, $|\pi| = k$, is a pair of sequences, $1 = c_1 \leq \cdots \leq c_{m+1} = k + 1$ of $m + 1$ distinct vertical lines (column divisions) and $1 = r_1 \leq \cdots \leq r_{n+1} = k+1$ of $n+1$ distinct horizontal lines (row divisions) which divide the plot of a permutation into cells, such that for each i, j points of π in cell ij are

- increasing, if $M_{ij} = 1$;
- decreasing, if $M_{ij} = -1$ or
- there are no points of π in ij, if $M_{ij} = 0$. [Vat11, VW11a]

Example. *An example of a* $0/\pm 1$ *matrix is*

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 1 & -1 & -1 \end{pmatrix}.$$

The permutation $\pi = 259836471$ *has an* M*-gridding, with column divisions* $c_1, c_2, c_3, c_4 = 1, 3, 7, 10$ *and row divisions* $r_1, r_2, r_3 = 1, 6, 10$*, which is shown in figure 1.6.*

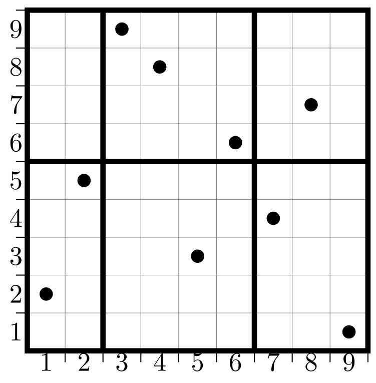


Figure 1.6: An M-gridding of $\pi = 259836471$.

Definitions. M*-griddable permutation,* M*-gridded permutation,* M*-griddable class, monotone grid class, standard figure, geometric grid class*

If π has an M-gridding, then π is said to be M*-griddable*. A permutation with such a gridding is called an M*-gridded permutation*. Similarly, a permutation class $\mathcal{C}$ is said to be M*-griddable* if every $\pi \in \mathcal{C}$ is M-griddable. The class consisting of all M-griddable permutations is called the *(monotone) grid class of* M, and is denoted Grid(M) [Vat11, Bri12, AAB$^+$11b].

We want to go further and define *geometric grid classes*, as through the nature of their definition they have a natural encoding.

Let M be a $0/\pm1$ matrix. The *standard figure* of M, denoted Λ, is the point set in $\mathbb{R}^2$ constrained to increasing line segments from $(i-1, j-1)$ to (i, j) if $M_{ij} = 1$, decreasing line segments from $(i-1, j)$ to $(i, j-1)$ if $M_{ij} = -1$, and no points between $(i-1, j-1)$ and (i, j) if $M_{ij} = 0$. The *geometric grid class* of M, denoted as $\mathrm{Geom}(M)$, is the set of permutations that can be drawn on the standard figure Λ as follows: choose n points in Λ where no two points lay on a common horizontal or vertical line, label these points from 1 to n from bottom to top, and read the labels off from left to right [ARV12, AAB+11b]. In other words, the plot of each permutation is a subset of Λ [Bev13].

Example. *Let the* $0/\pm1$ *matrix be*

$$M = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 1 \end{pmatrix}.$$

The permutation $\pi = 156324$ *is in* $\mathrm{Geom}(M)$.

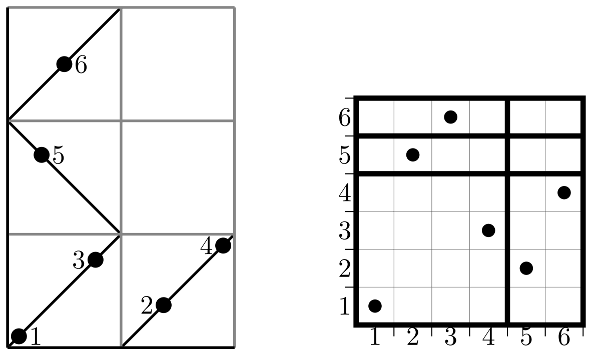


Figure 1.7: Standard figure of M, with π indicated, and the plot of π with grid lines.

In [AAB+11b] it has been noted that $\mathrm{Geom}(M) \subseteq \mathrm{Grid}(M)$, this is due to the fact that the points of the permutations in a geometric grid class are fixed on the lines in the plane, whereas in monotone grid classes the placement is more flexible.

Definitions. *cell graph*

The *cell graph* of a matrix M is a graph with set of vertices $\{(i, j) : M_{ij} \neq 0\}$, where vertices are adjacent if the corresponding cells of M share a row or a column.

Theorem 12 ([AAB+11b])**.** *If the cell graph of M is a forest then* $\mathrm{Geom}(M) = \mathrm{Grid}(M)$.

Example. *Let*

$$M = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

then its cell graph is

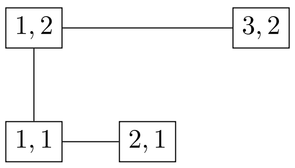


Figure 1.8: Example of the cell graph of a matrix.

Example. *Here is an example of a matrix M and a permutation π such that $\pi \in \mathrm{Grid}(M)$ but $\pi \notin \mathrm{Geom}(M)$. Let*

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \textit{ and } \pi = 1324.$$

The cell graph of M is

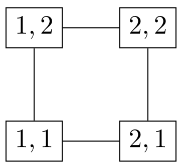


Figure 1.9: Cell graph of M.

and the geometric gridding of π is

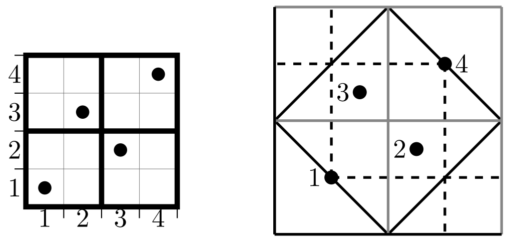


Figure 1.10: Plots of $\pi = 1324$.

The plot of π reveals that even though an M-gridding can be found for π it is not possible to find a geometrical gridding of π over M because if the points 1 and 4 are set on line segments in Λ, then moving the points 2 and 3 is not possible without moving 1 and 4 off their line segments. Thus

$$1324 \in \mathrm{Grid}(M) \textit{ but } 1324 \notin \mathrm{Geom}(M).$$

Theorem 13 ([AAB$^+$11b]). *Every geometrically griddable class is finitely based.*

Proposition 14 ([AAB$^+$11b]). *The union of a finite number of geometrically griddable classes is geometrically griddable.*

We can see again that having an understanding of smaller geometric grid classes allows us to construct larger classes.

Chapter 2

Encodings

Most research in permutation pattern classes was and still is interested in classifying and enumerating pattern classes. The cross-over from permutation pattern classes to formal languages came naturally and as there is more knowledge and computing power within languages. Overall the research into pattern classes is benefiting from this alternative representation.

In this chapter we will present three different encodings, namely the *rank encoding*, the *insertion encoding* and the *encoding of geometric grid classes*. Many permutation pattern classes lead to regular languages under the rank encoding [AAR03], which is a form of Lehmer code [Leh60]. Under the insertion encoding many pattern classes are either regular [Vat12] or context-free languages [ALR05]. All geometric grid classes have been found to be regular under their encoding [AAB$^+$11b]. Having regular and context-free languages makes pattern classes highly computable through their representation as regular expressions and automata. For more details on the theory of languages and automata see [HMU06]. We will not restate basic facts about languages and automata.

2.1 The Rank Encoding of Permutations

Definitions. *rank encoding, rank*

The *rank encoding* of a permutation $\pi = \pi(1)\pi(2)\ldots\pi(n)$ is the sequence $E_R(\pi) = p_1p_2\ldots p_n$, where for all $i \in \{1,\ldots,n\}$,

$$p_i = \left|\left\{j : j \in \{i,\ldots,n\}, \pi(j) \leq \pi(i)\right\}\right|$$

is the *rank* of $\pi(i)$, amongst the entries of π that have not occurred yet while reading the permutation π from left to right. [AAR03]. We denote the rank encoding of a permutation π as $E_R(\pi)$.

Example. *Let* $\pi = 541963728$ *then* $E_R(\pi) = 541632211$*. A step by step calculation is shown* *below.*

Permutation	*Encoding*	*Unused values*
541963728	∅	1234**5**6789
41963728	5	123**4**6789
1963728	54	**1**236789
963728	541	23678**9**
63728	5416	23**6**78
3728	54163	2**3**78
728	541632	2**7**8
28	5416322	**2**8
8	54163221	**8**
∅	541632211	∅

Reversely, the sequence $E_R(\sigma) = 314341221$ *represents the rank encoding of* $\sigma = 316582794$.

Encoding	*Permutation*	*Unused values*
314341221	∅	12**3**456789
14341221	3	**1**2456789
4341221	31	245**6**789
341221	316	24**5**789
41221	3165	247**8**9
1221	31658	**2**479
221	316582	4**7**9
21	3165827	4**9**
1	31658279	**4**
∅	316582794	∅

Definitions. *inversion*

The rank encoding is a natural encoding of the permutations sorted by networks and stacks, as introduced in [ALT97]. Another way of thinking of the rank encoding is as the language of inversions. An *inversion* in a sequence $s_1 \dots s_n$ is a subsequence $s_i s_j$, $i < j$, which is order isomorphic to 21. The *rank encoding* of a permutation π, $|\pi| = n$ is the sequence of the numbers of inversions (plus 1) that involves $\pi(i)$ of every factor of the form $\pi(i) \dots \pi(n)$, $i \in \{1, \dots, n\}$ of the permutation.

It is important to note for the characterisation of the language of rank encoded permutations that not every sequence in $\{1, \dots, n\}^n$ represents a permutation, as there are n^n sequences in $\{1, \dots, n\}^n$ and only $n!$ permutations of length n. For example the sequence 334644211 is the rank encoding of the permutation 346978215, whereas the sequence 234664311 cannot be decoded into a permutation as the second 6 in the encoding decodes to 10 but the decoded word will be of length 9. Additionally, we can see that the rank encoding is unique to every permutation and vice versa.

2.2 Classes, Token Passing Networks and the Rank Encoding

The abstract theory of *token passing networks* is essential to different aspects of computer science in real world situations. For example, a token passing network can represent a distributed computing network and the data communication within it or parallel computers and the bits and bytes flowing along the architecture. Feeding networks with numbered tokens and analysing the patterns of the output, has implications for the resequencing problem of packet-switching networks. Furthermore, the understanding of the outputs of token passing networks could lead to the improved apprehension of real world applications. The problem of sorting permutations using a stack, as specified by Knuth in [Knu97] can also be represented as an infinite token passing network.

Definitions. *token passing network (TPN)*

Formally, a *token passing network* is a finite directed graph with a designated input node and a designated output node. The input node has no incoming edges from other nodes whereas the output node has no outgoing edges to other nodes. The input node generates a sequence of tokens, labelled $1, 2, 3, \ldots$, and the output node collects the tokens in the order they arrive. These tokens are passed on to the nodes within the graph, where each node, apart from the input and output nodes, can hold at most one token at any time. The edges do not hold tokens but are there to pass them on.

The following must hold if a token t moves from a node x to a node y;

- There is an edge from x to y;
- x is the input node, and the tokens $1, \ldots, t-1$ have been moved, or x is any other node but not the output node;
- lastly, either y is the output node or y is not the input node and currently is not occupied by a token [ALT97].

Example. *The TPN described in Knuth's exercise in [Knu97] is based on an infinite stack. Figure 2.1 represents such a TPN with a finite stack of size* 3.

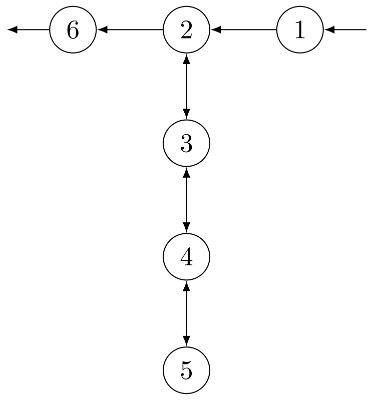


Figure 2.1: Single size 3 stack TPN.

A token passing network outputs permutations of the input sequence $1, \ldots, n$. The set of permutations output by a token passing network is closed under the containment order (or single point deletion) [ALR04].

Example. *The class of the TPN in figure 2.1 is* $Av(312)$*.*

The rank encoding of the permutations output by TPNs is a natural encoding. The rank of each token is the number of tokens still in the TPN (plus 1) which have smaller value than the currently output token. This is the reason for the name of this encoding. In general we know that the maximal rank of the language cannot exceed the number of internal (not including the input and output node) nodes of the network [ALR04].

Theorem 15. *[ALT97] The rank encoded class of output permutations of a TPN is regular.*

The non-minimal and non-deterministic automaton accepting the rank encoded language of a pattern class that is output by a TPN is built by letting each state of the automaton represent a possible configuration of tokens in the network [ALT97].

Example. *Let the token passing network be the single stack of size* 3 *as in figure 2.1. The network can be saturated with* 4 *tokens, so our maximal rank will be* 4*. The language of all words representing permutations created by the single stack of size* 3*, over the alphabet* $\Sigma = \{1, 2, 3, 4\}$ *is accepted by the minimal automaton below, the originally constructed non-deterministic automaton has* 32 *states.*

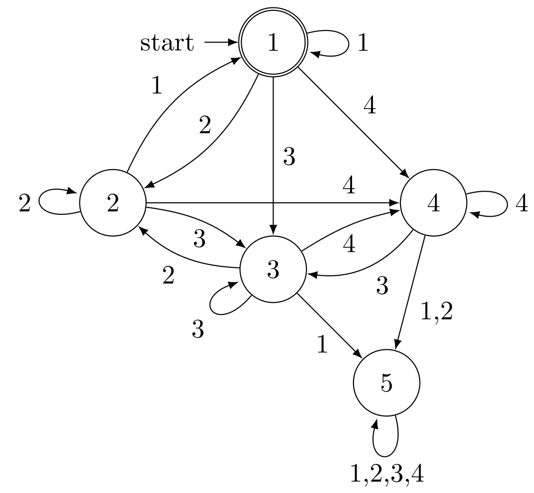


Figure 2.2: Minimal automaton accepting the rank encoded permutations from the TPN in figure 2.1.

The regular expression has the form

$$\left(\left((44^*3|3)\,(44^*3|3)^*\,2|2\right)\left((44^*3|3)\,(44^*3|3)^*\,2|2\right)^*\,1|1\right)^*$$

with the shortest words being

Encoding	*Permutation*
1	1
11	12
21	21
111	123
121	132
211	213
221	231
321	321
1111	1234
⋮	⋮

Definitions. *Ω_k, regular class*

In [AAR03] it was shown that the set of rank encoded permutations with highest rank k, denoted as $E_R(\Omega_k)$, is regular. If we have a pattern class that is a subset of Ω_k and a regular language under the rank encoding, we will call that pattern class a *regular class*. Further, in that same paper the following theorem was proven.

Theorem 16. *[AAR03] A closed subset of Ω_k is regular if and only if its basis is regular.*

The proof and the construction of the regular language of the basis is set around the theory of transducers, which translate one language to another using given rules. In this case the rules are based around point deletion in permutations and the effect point deletion has on the rank encoding.

Definitions. *finite state transducer*

A *finite state transducer* is a type of finite automaton with output strings. Thus it is a sextuple $(\Sigma, \Gamma, S, \delta, s_1, A)$, where Σ is the input alphabet, Γ is the output alphabet, S is the finite set of states, δ is the transition function $S \times (\Sigma \times \Gamma) \to S$, $s_1 \in S$ is the start state and $A \subseteq S$ is the set of accept states.

Example. *Below is an example of a transducer that takes words over the alphabet $\{1, 2\}$ and returns words over the same alphabet.*

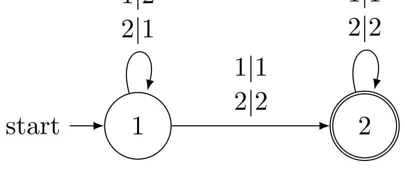


Figure 2.3: Example of a transducer.

This transducer takes a word, changes the letters until an unspecified index, from which on the letters are the same as in the input word. Applying the transducer to $w = 121212$ can return any word of the set $\{121212, 221212, 211212, 212212, 212112, 212122, 212121\}$.

First let us look at the construction of the language of the basis from the class. Let $\mathcal{C}$ be a regular class under the rank encoding with the language $E_R(\mathcal{C})$. We want to find the basis B of $\mathcal{C}$ which is the minimal set of permutations not in the class. In [AAR03] it is shown that the language $E_R(B)$ of B can be found using the equation

$$E_R(B) = (E_R(\mathcal{C}))^C \cap ((E_R(\mathcal{C}))^C \mathcal{D}^t)^C,$$

where $\mathcal{D}$ is a transducer that deletes an arbitrary letter in a rank encoded permutation, and returns a word that represents the permutation that had the point removed that corresponds to the removed letter. Further, $\mathcal{D}^t$ is the transpose of the transducer $\mathcal{D}$, which means that the transducer $\mathcal{D}^t$ will add a letter to the word, because the transpose of a transducer has the input and output alphabets interchanged, as well as the letters on the transitions.

Example. *Let $k = 3$ then the one point deletion transducer $\mathcal{D}$ has input alphabet $\{1, 2, 3\}$, output alphabet $\{\varepsilon, 1, 2, 3\}$ and the following form.*

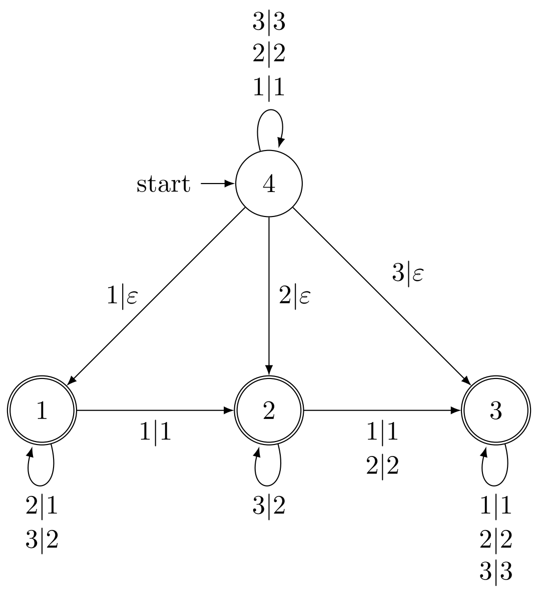


Figure 2.4: One point deletion transducer over the alphabets $\Sigma = \{1, 2, 3\}$, $\Gamma = \{\varepsilon, 1, 2, 3\}$.

So if the permutation $\pi = 243516$ with the encoding $E_R(\pi) = 232211$ has the point $\pi(3) = 3$ point deleted, then the output permutation is $\pi' = 23415$ with the encoding $E_R(\pi') = 22211$. The one point deletion transducer does exactly that, but without knowing what the underlying permutation is. So it takes the word $w = 232211$ and returns a set of words, which are all valid rank encodings with a letter less than the starting word and $w' = 22211$ is amongst them.

Just to re-iterate, to transpose a transducer $\mathcal{D}^t$ means to change the letters on the transitions, so the output letter is turned to the input letter and vice versa.

Example. *Figure 2.4 shows the deletion transducer over the input alphabet $\{1, 2, 3\}$, whereas the following figure shows its transpose.*

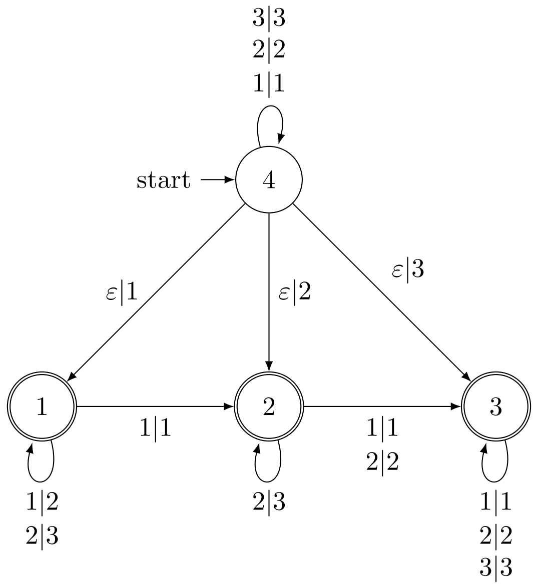


Figure 2.5: Transpose of the one point deletion transducer, which is now a transducer over the input alphabet $\{\varepsilon, 1, 2, 3\}$ and output alphabet $\{1, 2, 3\}$.

This means that the transducer now adds a letter to a rank encoded word, and thus a point to the corresponding permutation. Take $w = 22211$ *and add a letter to get* $w' = 232211$*. The corresponding permutations are* $E_R(\pi) = w = 23415$ *and* $E_R(\pi') = w' = 243516$*.*

Secondly, it has been shown in [AAR03] that it is also possible to move from knowing the language of the basis to the language of the class under the rank encoding. This is done by using an involvement transducer $\mathcal{H}$ that removes any number of letters from the input word while still returning a valid rank encoded word which corresponds to the permutation with the same set of points removed.

$$E_R(\mathcal{C}) = (E_R(B)\mathcal{H}^t)^C \cap E_R(\Omega_k).$$

Example. *The involvement transducer for* $k = 3$ *has the following form.*

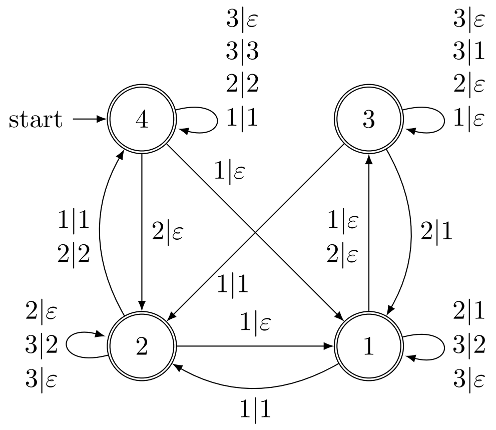


Figure 2.6: Diagram of involvement transducer $\mathcal{H}$ for $k = 3$.

Example. *Let us take the TPN as represented in figure 2.7. The language of the permutation* *class $\mathcal{C}$ of that TPN is regular under the rank encoding. So is the language of the basis B, where the basis is infinite.*

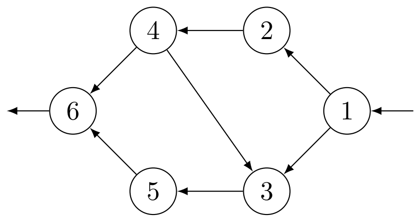


Figure 2.7: A TPN with infinite but regular basis.

The language of the class is

$$E_R(\mathcal{C}) = \left((22^*3|3)\,(12^*3|3)^*\,\left(12^*1|2\,\left((33^*1|33^*2|2)\,(33^*1|2)^*\,1|1\right)\right)|22^*1|1\right)^*$$

and the language of the basis is

$$E_R(B) = 31(31)^*321|322321.$$

The shortest words of these languages are

Class

Encoding	*Permutation*
1	1
11	12
21	21
111	123
211	213
121	132
⋮	⋮

Basis

Encoding	*Permutation*
322321	324651
3231321	3251764
323131321	325174986
⋮	⋮

The two figures below show the automata accepting these languages.

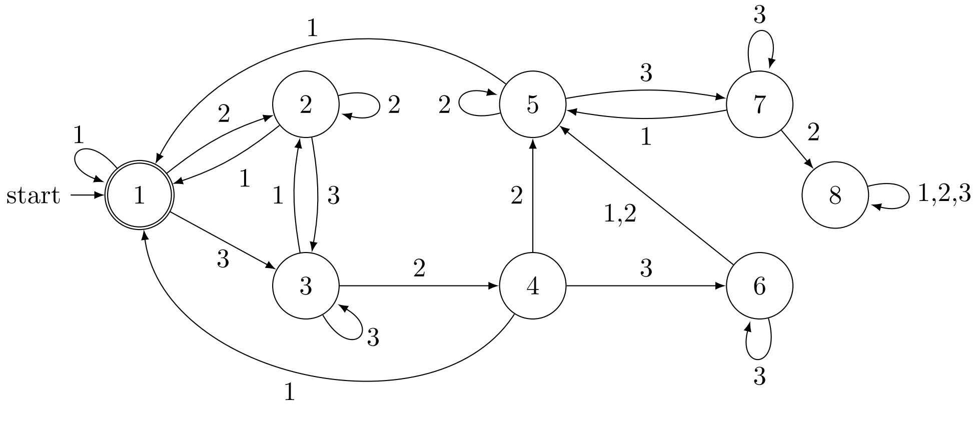


Figure 2.8: Deterministic automaton accepting the language of the class.

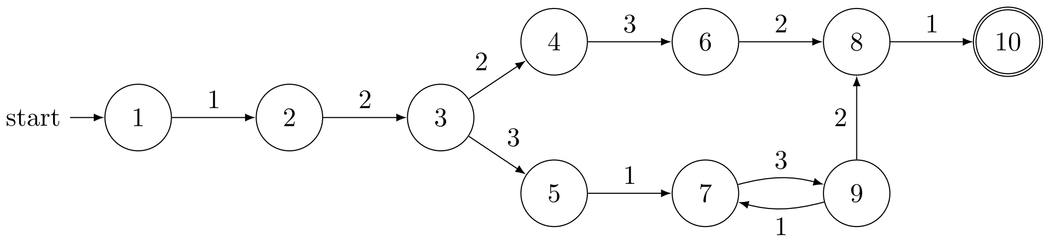


Figure 2.9: Non-deterministic automaton accepting the language of the basis.

It is interesting to see that the automaton of the basis, although larger is less complicated than the automaton of the language. This is similar in the regular expressions of these languages.

2.3 Insertion Encoding of Permutations

Definitions. *insertion encoding, configuration, slot*

For completeness we mention a generalisation of the rank encoding. The idea of the *insertion encoding* is to keep track of how a permutation is built by adding the next highest element into a configuration of the permutation containing slots [Eld04]. A *configuration* is the state of a permutation after adding a maximal element. It is represented as a sequence of numbers and *slots*, denoted by ␣. The manner of the insertion is recorded and builds the insertion encoded word. There are four different ways of inserting a new element x into a slot:

␣ → ␣ x ␣	is represented by **m**, for middle,
␣ → x ␣	is represented by **l**, for left,
␣ → ␣ x	is represented by **r**, for right,
␣ → x	is represented by **f**, for fill.

Each of these insertion operations carries a subscript that indicates on which slot in the current configuration it operates [ALR05, SV09]. The alphabet of the insertion encoding will be those operations with their subscripts and the words over that alphabet will be the instructions how to construct the corresponding permutation. We denote the insertion encoding of a permutation π as $E_I(\pi)$.

Example. *The permutation* $\pi = 316582794$ *is insertion encoded as* $E_I(\pi) = m_1m_2f_1r_2m_1f_1l_2f_1f_1$.

Configuration	*Encoding*
␣	
␣ 1 ␣	m_1
␣ 1 ␣ 2 ␣	m_1m_2
31 ␣ 2 ␣	$m_1m_2f_1$
31 ␣ 2 ␣ 4	$m_1m_2f_1r_2$
31 ␣ 5 ␣ 2 ␣ 4	$m_1m_2f_1r_2m_1$
3165 ␣ 2 ␣ 4	$m_1m_2f_1r_2m_1f_1$
3165 ␣ 27 ␣ 4	$m_1m_2f_1r_2m_1f_1l_2$
3165827 ␣ 4	$m_1m_2f_1r_2m_1f_1l_2f_1$
316582794	$m_1m_2f_1r_2m_1f_1l_2f_1f_1$

Conversely, $m_1r_2m_2m_1f_4f_3f_2f_1$ *is decoded to the permutation* 84716352.

Encoding	*Configuration*
$m_1r_2m_2m_1f_4f_3f_2f_1$	␣
$r_2m_2m_1f_4f_3f_2f_1$	␣ 1 ␣
$m_2m_1f_4f_3f_2f_1$	␣ 1 ␣ 2
$m_1f_4f_3f_2f_1$	␣ 1 ␣ 3 ␣ 2
$f_4f_3f_2f_1$	␣ 4 ␣ 1 ␣ 3 ␣ 2
$f_3f_2f_1$	␣ 4 ␣ 1 ␣ 352
f_2f_1	␣ 4 ␣ 16352
f_1	␣ 4716352
	84716352

The language of a set of permutations under the insertion encoding will be regular if we limit the subscript of the letters to at most k. In other words, we are limiting the depth of the filling of the open slots in each configuration [Vat12].

A set of permutations is context-free under the insertion encoding if we let the push-down automaton accepting the language be such that each transition of the automaton must correspond to some single letter of the insertion encoding and the number of symbols in the stack is equivalent to the number of slots available after the prefix of the word has been interpreted [ALR05].

2.4 Regular Insertion Encoded Pattern Classes

Let us observe that it is possible to get a regular language of permutations under the insertion encoding, by limiting the number of slots available at any configuration of the permutation. We will denote this insertion encoding by E_{IR} .

Definitions. *slot bounded permutation*

For each positive integer k the set $\mathcal{SB}(k)$ of permutations for which the insertion encoding never includes more than k slots is called the set of k *slot bounded permutations* [ALR05].

Proposition 17. *[ALR05] For each positive integer k the set $\mathcal{SB}(k)$ is a pattern class. Its basis consists of the $(k+1)!k!$ permutations of length $2k+1$ of the form babab ... bab where the positions marked with b are occupied by the numbers $\{k+1, k+2, \dots, 2k+1\}$ while those marked by a are occupied be the numbers $\{1, 2, \dots, k\}$.*

Theorem 18. *[ALR05] Let $\mathcal{C}$ be a pattern class that is a subclass of $\mathcal{SB}(k)$ for some k. The following are equivalent:*

- *The language $E_{IR}(\mathcal{C})$ is regular.*
- *There is a regular language $E_{IR}(B)$ defining a subset of $B \subseteq \mathcal{SB}(k)$ such that $\mathcal{C} = Av(B) \cap \mathcal{SB}(k)$.*

In fact B can, but need not, be chosen to consist of those elements of the basis of $\mathcal{C}$ which belong to $\mathcal{SB}(k)$, and there is an effective procedure for passing from the language of B to that of $\mathcal{C}$ and vice versa.

Encoding permutations with this version of the insertion encoding is more inefficient than using the rank encoding and does not lead to different or improved results. Thus, so far the rank encoding has been commonly used instead, when wanting to look at permutation pattern classes with regular languages.

2.5 Context Free Insertion Encoded Pattern Classes

To reach context free pattern classes with the insertion encoding, the subscripts on the slot operations will stay limited but we will allow for counting slots from the right additionally to counting from the left side. The counting direction will be distinguished through the negative sign in the index. We will denote this context free insertion encoding by E_{ICF}.

Definitions. *insertion bounded class*

Let k be a positive integer. The set $\mathcal{IB}(k)$ consists of all those permutations whose insertion encodings can be written using only operations whose subscripts come from $\{\pm 1, \pm 2, \ldots, \pm k\}$. We call this the *insertion bounded class of depth k.* [ALR05]

Example. *The permutation $\pi = 2413657$ can be encoded as $E_{ICF}(\pi) = m_1 l_1 l_{-1} f_1 m_{-1} f_1 f_1$.*

Configuration	*Encoding*
␣	
␣ 1 ␣	m_1
2 ␣ 1 ␣	$m_1 l_1$
2 ␣ 13 ␣	$m_1 l_1 l_{-1}$
2413 ␣	$m_1 l_1 l_{-1} f_1$
2413 ␣ 5 ␣	$m_1 l_1 l_{-1} f_1 m_{-1}$
241365 ␣	$m_1 l_1 l_{-1} f_1 m_{-1} f_1$
2413657	$m_1 l_1 l_{-1} f_1 m_{-1} f_1 f_1$

So $\pi \in \mathcal{IB}(1)$.

Proposition 19. *[ALR05] Each set $E_{ICF}(\mathcal{IB}(k))$ is a context free pattern class. Its basis consists of the set of permutations of the form:*

$$c_1 a_1 c_2 a_2 \ldots c_k a_k (2k+1) a_{k+1} c_{k+1} \ldots a_{2k} c_{2k}$$

where $\{a_1, \ldots, a_{2k}\} = \{1, \ldots, 2k\}$ and $\{c_1, \ldots, c_{2k}\} = \{2k+2, 2k+3, \ldots, 4k+1\}$.

Theorem 20. *[ALR05] Any context free class is a subclass of $\mathcal{IB}(k)$ for some k.*

2.6 Geometric Grid Class Encoding

Definitions. *refinement*

Before we can talk about the encoding of geometric grid classes, we have to introduce a couple of definitions on the $0, \pm 1$ matrices. Let M be a $0, \pm 1$ matrix of size $m \times n$. As mentioned in section 1.5 we are indexing matrices in a Cartesian coordinate fashion. The *refinement* $M^{\times q}$ is the

$mq \times nq$ matrix obtained by replacing every 0 entry of M by a $q \times q$ zero submatrix, every 1 entry of M by a $q \times q$ submatrix of the form

$$\begin{matrix} 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \iddots & \vdots \\ 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{matrix}$$

and every -1 entry of M is replaced by a $q \times q$ submatrix of the form

$$\begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{matrix}$$

Example. *The refinement $M^{\times 2}$ of $M = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ is*

$$M^{\times 2} = \begin{pmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Definitions. *partial multiplication matrix (PMM), column and row signs*

The matrix M is a *partial multiplication matrix (PMM)* if it has *column* and *row signs* $c_1, \ldots, c_m, r_1, \ldots, r_n \in \{1, -1\}$ such that $M_{ij} = c_i r_j$ or $M_{ij} = 0$. In [AAB$^+$11b] it is shown that for every $0, \pm 1$ matrix M, its refinement $M^{\times 2}$ is a PMM. Furthermore there it is also proven that every geometric grid class is a geometric grid class of a PMM. We can thus assume that from now on our matrix M is a PMM, if it is not we can replace M with its refinement $M^{\times 2}$.

Example. *The matrix $M = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ has column and row signs $c_1 = r_2 = -1,\ c_2 = c_3 = r_1 = 1$.*

Definitions. *cell alphabet*

We can now introduce the language of geometrically griddable permutations. Let M be a $m \times n$ PMM with column and row signs $c_1, \ldots, c_m, r_1, \ldots, r_n$. The *cell alphabet* of M is $\Sigma = \{a_{ij} : M_{ij} \neq 0\}$. Any word in Σ^* describes a permutation that is geometrically griddable by M by following the rules below of how to construct permutations from these words. Each letter of Σ describes the cell that the point is placed into, the order we place these points into the cells is defined by the column and row signs and is described in table 2.1 [VW11a].

c_i	r_j	Order of insertion	
1	1	left to right and bottom to top	$\nearrow$
1	-1	left to right and top to bottom	$\searrow$
-1	1	right to left and bottom to top	$\nwarrow$
-1	-1	right to left and top to bottom	$\swarrow$

Table 2.1: Procedure of adding points into cells based on the column and row signs.

Example. *Take* $M = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ *with column and row signs* $c_1 = r_2 = -1,\ c_2 = c_3 = r_1 = 1$ *and* $w = a_{12}a_{21}a_{21}a_{32}a_{11}a_{12}a_{11}a_{12}$ *over the cell alphabet of* M*. Then the permutation that is encoded by* w *and is geometrically griddable by* M *is* $\pi = 54638127$*.*

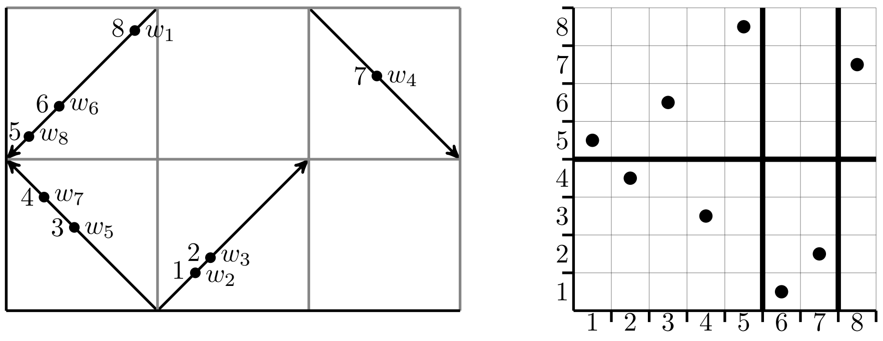


Figure 2.10: Gridded figure and plot of $E_{G(M)}(\pi) = w = a_{12}a_{21}a_{21}a_{32}a_{11}a_{12}a_{11}a_{12}$, where $\pi =$ 54638127.

We will denote this encoding by $E_{G(M)}(\pi)$. It is worth noting that the whole language Σ^* over the cell alphabet Σ, is the regular language of the geometric grid class of M. But there are multiple encoded words that correspond to the same permutation. This non-uniqueness is discussed in detail in section 5.

Chapter 3

Languages of Sets of Permutations and Classes

In this chapter we will be concentrating on the rank encoding solely. So when we talk about the encoding, we mean the rank encoding and that the language is regular under the rank encoding. Further, unless otherwise stated in this chapter we will be denoting the rank encoding of a permutation π as $E(\pi)$.

We will be looking at the languages constructed by plus- and minus-(in)decomposable permutations, σ-decomposable permutations, and as there are equivalent definitions for classes, we will be also investigating the languages of the direct and skew sum of classes and the wreath product of classes. Further we will be proving that the language of simple permutations is regular under the rank encoding and that we can find the set of simple permutations in a non-regular class. First we will recall some definitions and vital results.

3.1 Introduction

As mentioned before, we denote the set of permutations with maximum rank $k \in \mathbb{N}$ as Ω_k. In [AAR03] Albert, Atkinson and Ruškuc determined that $E(\Omega_k)$ is a regular language.

Definitions. *interval, block*

Furthermore we re-introduce the concept of block-decomposition of permutations. An *interval* (or *block* see [AA05]) in a permutation σ is a factor of contiguous values of σ such that their indices are consecutive.

Example. *In the permutation* $\pi = 346978215$*,* $\pi(4)\pi(5)\pi(6) = 978$ *is an interval, whereas* $\pi(1)\pi(2)\pi(3)\pi(4) = 3469$ *is not.*

Definitions. *simple permutation*

It is easy to see that every permutation of length n has intervals of length 0, 1 and n, at least. The permutations of length n that *only* contain intervals of length 0, 1 and n are said to be *simple* [Bri10].

Example. *The permutation* $\pi = 346978215$ *is not simple as we have seen in the example above that it contains an interval, on the other hand* $\sigma = 526184937$ *is simple as there are no intervals* *of length strictly greater than* 1*, except the whole of* σ*. Figure 3.1 shows* π *and* σ*, with non-trivial intervals indicated.*

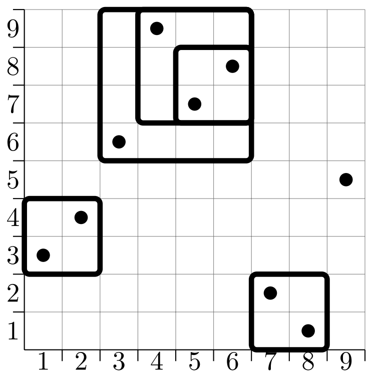
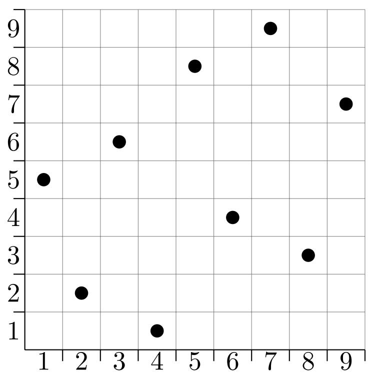



Figure 3.1: Plots of 346978215 and 526184937, with all non-trivial intervals indicated.

Definitions. *inflation, block-decomposition, deflation,* σ*-decomposable*

Recall that, given a permutation σ of length m and non-empty permutations $\alpha_1, \ldots, \alpha_m$, the *inflation* of σ by $\alpha_1, \ldots, \alpha_m$, written as $\sigma[\alpha_1, \ldots, \alpha_m]$, is the permutation obtained by replacing each entry $\sigma(i)$ by an interval that is order isomorphic to α_i, where the relative ordering of the intervals corresponds to the ordering of the entries of σ [AA05]. Conversely a *block-decomposition* or *deflation* [AA05] of π is any expression of π as an inflation $\pi = \sigma[\alpha_1, \ldots, \alpha_m]$. We say that a permutation π is σ*-decomposable* if $\pi = \sigma[\alpha_1, \ldots, \alpha_n]$. For notation, we will say that $\pi_i = \pi(x) \ldots \pi(y)$ is the contiguous subsequence of $\pi = \sigma[\alpha_1, \ldots, \alpha_n]$ corresponding to the block α_i. Additionally, we will use the same notation for the blocks π_i of π in its plot. In the rank encoding of π, $E(\pi)$, we will identify the subword corresponding to $\pi_i = \pi(x) \ldots \pi(y)$ as $E(\pi)[x, \ldots, y]$.

Example. *The inflation of* $\sigma = 25413$ *with* $\alpha_1 = 12$, $\alpha_2 = 1$, $\alpha_3 = 12$, $\alpha_4 = 321$, $\alpha_5 = 1$ *is* $25413[12, 1, 12, 321, 1] = 459783216$ *and alternatively a possible block-decomposition of* 459783216 *is* $25413[12, 1, 12, 321, 1]$. *This inflation and decomposition are shown in figure 3.2.*

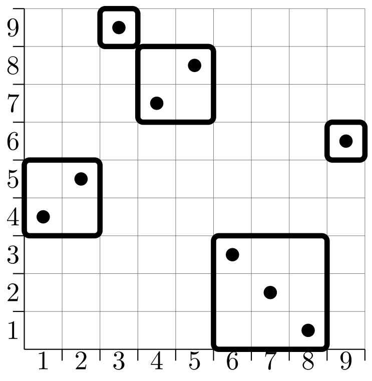


Figure 3.2: Plot of inflation $25413[12, 1, 12, 321, 1] = 459783216$.

This decomposition is not unique for π. However, Albert and Atkinson proved:

Proposition 21 ([AA05])**.** *Let* π *be a permutation of finite length greater than* 1*. There is a unique simple finite permutation* σ*, with* $|\sigma| > 1$*, and a sequence* $\alpha_1, \ldots, \alpha_n$ *of non-empty permutations* *such that*

$$\pi = \sigma[\alpha_1, \dots, \alpha_n].$$

If $\sigma \neq 12, 21$ then $\alpha_1, \dots, \alpha_n$ are also uniquely determined by π. If $\sigma = 12$ or 21, then α_1, α_2 are unique so long as we require that α_1 is plus-indecomposable or minus-indecomposable, respectively.

Example. *Utilising the above proposition, the unique block-decomposition of* 459783216 *with a simple permutation is given by* 2413[12, 312, 321, 1] *as shown in figure 3.3.*

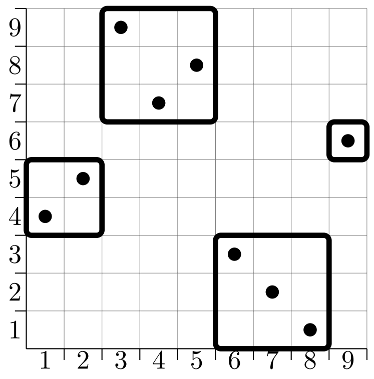


Figure 3.3: Plot of unique block-decomposition of 459783216 = 2413[12, 312, 321, 1] as defined by Proposition 21, where 2413 is simple.

Definitions. ε, $\mathcal{L}^{(n)}$, $x + \mathcal{L}$, $\mathcal{L}|_1^n$

Lastly, let us introduce some language theoretic notation that will be used. We write ε for the empty word. If $\mathcal{L}$ is any language then $\mathcal{L}^{(n)}$ is the set of words of $\mathcal{L}$ up to length $n \in \mathbb{N}$, and as the set is finite, it is regular.

Furthermore, if $\mathcal{L} \subseteq \{l_1, \dots, l_n\}^*$ is any language, then we define $x+\mathcal{L}$, $x \in \mathbb{N}$, to be a language over the alphabet $\{l_1 + x, \dots, l_n + x : l_1, \dots, l_n \in \mathcal{L}\}$, $x + \mathcal{L} \subseteq \{l_1 + x, \dots, l_n + x\}^*$.

We observe that this new language is still regular if $\mathcal{L}$ is regular and say that the language $\mathcal{L}$ is shifted upwards by x.

Lemma 22. *If $\mathcal{L}$ is a regular language then so is $x + \mathcal{L}$.*

Proof. Let $\mathscr{A}(\mathcal{L})$ be the automaton accepting the regular language $\mathcal{L}$. We can construct the automaton $\mathscr{A}(x+\mathcal{L})$ by replacing the letters $l_1, \dots, l_n$ in the transitions of $\mathscr{A}(\mathcal{L})$ with the corresponding letters $l_1 + x, \dots, l_n + x$. Clearly the resulting automaton accepts the language $x + \mathcal{L}$. □

Let us say that the union of all possible shifts up to n of a regular language $\mathcal{L}$, is

$$\mathcal{L}|_1^n = \bigcup_{i=1}^{n} i + \mathcal{L}$$

which is a regular language as regularity is preserved under a finite number of unions.

3.2 Plus-Decomposable and Plus-Indecomposable Permutations

Definitions. *plus-decomposable, direct sum of permutations, plus-indecomposable*

One of the special cases in proposition 21 is the block-decomposition with $\sigma = 12$. A permutation π is said to be *plus-decomposable* (or is a *direct sum* of α_1 and α_2) if it can be written in the block-decomposition as

$$\pi = 12[\alpha_1, \alpha_2].$$

Conversely, we call a permutation *plus-indecomposable* if it has no plus-decomposition [AA05].

In general, α_1 and α_2 are not unique, but if we require α_1 to be plus-indecomposable, both α_1 and α_2 are unique to π.

Example. *The permutation* 21436875 *is plus-decomposable. A possible decomposition is* $21436875 = 12[2143, 2431]$*, but that is not unique, as* 2143 *is plus-decomposable. Thus, the plus-decomposition of* 21436875 *with unique* α_1 *and* α_2 *is* $21[21, 214653]$.

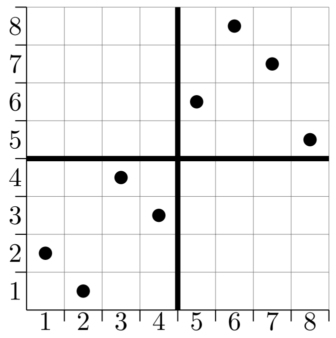
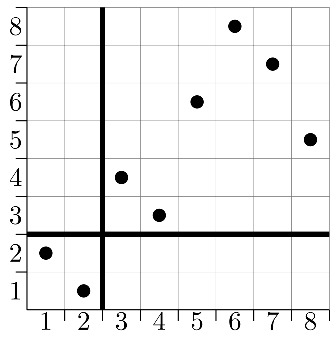



Figure 3.4: Two plus-decompositions of 21436875, the right one being the decomposition with α_1 plus-indecomposable.

The following lemma is a characterisation of plus-decomposable permutations, to outline the form these permutations have. It additionally facilitates the characterisation of plus-indecomposable permutations.

Lemma 23. *Let* $\pi = \pi(1) \dots \pi(n)$ *be a permutation, then the following are equivalent:*

1. $\pi = 12[\alpha_1, \alpha_2]$ *is plus-decomposable with* $|\alpha_1| = \ell$ *and* $|\alpha_2| = n - \ell$ *for* $\ell \in \mathbb{N} \setminus \{0\},\ \ell < n$.
2. $E(\pi) = E(\pi)[1, \dots, \ell, \ell+1, \dots, n] = E(\pi)[1, \dots, \ell]E(\pi)[\ell+1, \dots, n] = E(\alpha_1)E(\alpha_2),\ \ell \in \mathbb{N} \setminus \{0\},\ \ell < n$.
3. $\pi = \eta\tau$*, where* $\eta = \pi(1) \dots \pi(\ell)$ *is a permutation of* $\{1, \dots, \ell\}$ *and* τ *is a permutation of* $\{\ell+1, \dots, n\}$, $\ell \in \mathbb{N} \setminus \{0\},\ \ell < n$.

Proof. First we will show that point 1 implies point 3. So let $\pi = 12[\alpha_1, \alpha_2]$ be plus-decomposable with $|\alpha_1| = \ell$ and $|\alpha_2| = n - \ell$ for $\ell \in \mathbb{N} \setminus \{0\},\ \ell < n$. Then by the definition of block-decompositions of permutations, π consists of the concatenation of two intervals A_1 and A_2, where A_1 is a contiguous sequence order isomorphic to α_1 and A_2 is a contiguous sequence order isomorphic to α_2. Further, the relative ordering of A_1 and A_2 corresponds to the permutation 12. Thus A_1 is a sequence over $\{1, \dots, \ell\}$ order isomorphic to α_1 and A_2 is a sequence over $\{\ell+1, \dots, n\}$ order isomorphic to α_2. In other words, $A_1 = \eta$ is a permutation of length ℓ and $A_2 = \tau$ is a permutation of $\{\ell+1, \dots, n\}$, and $\pi = A_1A_2 = \eta\tau$.

Next, let us show that point 3 implies point 2. Let $\pi = \eta\tau$, where $\eta = \pi(1)\dots\pi(\ell)$ is a permutation of $\{1,\dots,\ell\}$ and τ is a permutation of $\{\ell+1,\dots,n\}$. Then the encoding of π is

$$E(\pi) = E(\eta\tau) = E(\eta)E(\tau).$$

This is because η is a permutation, hence a closed interval and thus has a valid rank encoding, which is concatenated with the encoding of τ, as the rank encoding of two order isomorphic sequences is the same.

Finally, we will prove that point 2 implies 1. Let $E(\pi) = E(\pi)[1,\dots,\ell,\ell+1,\dots,n] = E(\pi)[1,\dots,\ell]E(\pi)[\ell+1,\dots,n] = E(\alpha_1)E(\alpha_2)$. Then π consists of the concatenation of two contiguous sequences, a_1 a sequence of $\{1,\dots,\ell\}$ order isomorphic to α_1 and a_2 a sequence of $\{\ell+1,\dots,n\}$ order isomorphic to α_2. We can see that every letter in a_1 is less than any letter in a_2. Thus giving us the relative ordering of a_1 and a_2 corresponding to 12. Thus π has a block-decomposition of the form $\pi = 12[\alpha_1, \alpha_2]$, with $|\alpha_1| = \ell$ and $|\alpha_2| = n - \ell$. □

To prove that the set of plus-decomposable permutations and the set of plus-indecomposable permutations of a regular pattern class $\mathcal{C}$ are regular under the rank encoding, we will first prove that the subset of plus-indecomposable permutations of a regular pattern class $\mathcal{C}$ form a regular language under the rank encoding. Then it will follow that the complement set of plus-decomposable rank encoded permutations is also regular, since the family of regular languages are closed under complement. The proof considers the automaton that accepts the regular class $\mathcal{C}$ and modifies it to accept only plus-indecomposable rank encoded permutations.

Theorem 24. *Let $\mathcal{C}$ be a regular class. Then $\mathcal{I}_P(\mathcal{C})$, the set of all plus-indecomposable permutations of $\mathcal{C}$, is also regular under the rank encoding.*

Proof. Conversely to the characterisation in lemma 23, the rank encoding $E(\pi)$ of a plus--indecomposable permutation π of length n never contains an initial segment of the form $E(\pi)[1\dots\ell] = E(p)$, where $\ell < n$ and p is a permutation of length ℓ.

We will utilise this description to construct the automaton accepting the language of the set of plus-indecomposable permutations under the rank encoding.

The automaton accepting $E(\mathcal{I}_P(\mathcal{C}))$ is based on the unique minimal automaton of $E(\mathcal{C})$. Let that automaton be

$$\mathscr{A} = \big(\Sigma,\, S,\, \delta,\, s_1,\, A\big),$$

where Σ is the alphabet, S the set of states, s_1 the start state, A the set of accept states and $\delta : S \times \Sigma \to S$ the transition function.

We will construct the automaton accepting only the plus-indecomposable rank encoded permutations as follows

$$\mathscr{I}_P = \big(\Sigma,\, S \cup \{x,y\},\, \delta',\, x,\, A \cup \{x\}\big),$$

where x and y are new states and $\delta' : (S \cup \{x,y\}) \times \Sigma \to S \cup \{x,y\}$ is a new transition function defined as:

$$\delta'(y,\alpha) = y \qquad\qquad \delta'(x,\alpha) = \delta(s_1,\alpha)$$
$$\delta'(s_a,\alpha) = y \qquad\qquad \delta'(s,\alpha) = \delta(s,\alpha)$$

for all $\alpha \in \Sigma$; $s \in S \setminus A$; $s_a \in A$.

In $\mathscr{I}_P$ the new states x, y are the new start state and sink state, respectively. A sink state is a state q, where q is not the start state and $q \notin A$, and the transition δ from q for any letter $\alpha \in \Sigma$ is $\delta(q, \alpha) = q$. We are introducing a new start state, to avoid the resulting language being empty, in case the original start state s_1 is an accept state.

Let us now show that the new automaton $\mathscr{I}_P$ indeed only accepts the rank encodings corresponding to plus-indecomposable permutations from the automaton $\mathscr{A}$ of the regular class $\mathcal{C}$. Let w be a word accepted by $\mathscr{A}$. If we end up in an accept state of $\mathscr{A}$ before reading the entire word, so there is an initial segment in w that is a rank encoding of a permutation in $\mathcal{C}$, then the new transition function δ' will send us to the sink state y and the word w will not be accepted by $\mathscr{I}_P$. On the other hand if w is a word accepted by $\mathscr{A}$ and by $\mathscr{I}_P$, we end up in an accept state only when the entire word w is read. The first case is only possible for plus-decomposable permutations, as shown in lemma 23.

The automaton $\mathscr{I}_P$ only accepts the words corresponding to plus-indecomposable permutations of $\mathcal{C}$ under the rank encoding. Thus the language $E(\mathcal{I}_P(\mathcal{C}))$ is regular, which concludes the proof of Theorem 24. □

Corollary 25. *Let $\mathcal{C}$ be a regular class. Then $\mathcal{D}_P(\mathcal{C})$, the set of all plus-decomposable permutations of $\mathcal{C}$, is also regular under the rank encoding.*

Proof. Let $\mathcal{I}_P(\mathcal{C}) \subseteq \mathcal{C}$ be the set of all plus-indecomposable permutations in $\mathcal{C}$.

As $\mathcal{D}_P(\mathcal{C})$ and $\mathcal{I}_P(\mathcal{C})$ are complementary in $\mathcal{C}$ we have

$$\mathcal{D}_P(\mathcal{C}) = \mathcal{C} \backslash \mathcal{I}_P(\mathcal{C}) \Rightarrow E(\mathcal{D}_P(\mathcal{C})) = E(\mathcal{C}) \cap E(\mathcal{I}_P(\mathcal{C}))^C.$$

As regular languages are closed under intersection and complement, $E(\mathcal{D}_P(\mathcal{C}))$ is regular. □

In summary we have shown that the subsets of plus-decomposable and plus-indecomposable permutations of a regular class are also regular languages under the rank encoding. Next let us see whether the same is true for minus-(in)decomposable permutations.

3.3 Minus-Decomposable and Minus-Indecomposable Permutations

Definitions. *minus-decomposable, skew sum of permutations, minus-indecomposable*

The other special case in proposition 21 is the block-decomposition with $\sigma = 21$. We say that a permutation π is *minus-decomposable* (or is a *skew sum* of α_1 and α_2) if it can be written in the block-decomposition as

$$\pi = 21[\alpha_1, \alpha_2].$$

Conversely we say that a permutation is *minus-indecomposable* if it has no minus-decomposition. The decomposition of a minus-decomposable permutation is unique, if α_1 is assumed to be minus-indecomposable.

Example. *The permutation* 86735241 *is minus-decomposable as for example* 21[312, 35241]*, but the unique decomposition of* 86735241*, where the first subpermutation is minus-indecomposable, is* 21[1, 6735241].

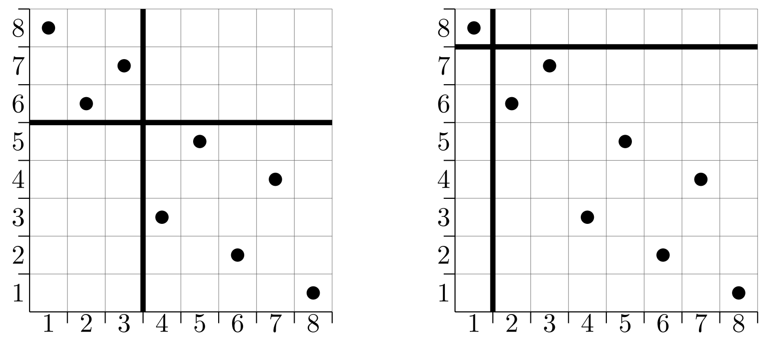


Figure 3.5: Two minus-decompositions of 86735241, the right plot showing the unique decomposition with the first interval being minus-indecomposable.

In a regular pattern class it is simpler to deal with the language that describes the rank encoding of minus-decomposable permutations rather than the minus-indecomposable permutations

Theorem 26. *Let $\mathcal{C}$ be a regular class. Then $\mathcal{D}_M(\mathcal{C})$, the set of all minus-decomposable permutations of $\mathcal{C}$, is also regular under the rank encoding.*

Proof. Let $E(\mathcal{C})$ be the regular language of $\mathcal{C}$ under the rank encoding where the alphabet of $E(\mathcal{C})$ is $\{1, \ldots, k\}$, $k \in \mathbb{N}$ and let $E(\mathcal{D}_M(\mathcal{C}))$ be the rank encoded language of $\mathcal{D}_M(\mathcal{C})$.

Let $\pi \in \mathcal{D}_M(\mathcal{C})$ be arbitrary with $\pi = 21[\alpha_1, \alpha_2]$, where $|\pi| = n$ and $|\alpha_2| = d < k$. We know that $d < k$ as otherwise the rank of the elements in π corresponding to α_1 will exceed k, contradicting that $\mathcal{D}_M(\mathcal{C}) \subseteq \mathcal{C} \subseteq \Omega_k$.

Then from Figure 3.6 we see that

$$E(\pi) = E\big(\pi(1) \ldots \pi(n-d)\ \pi(n-d+1) \ldots \pi(n)\big) = p_1 \ldots p_{n-d}\ p_{n-d+1} \cdots p_n,$$

where $p_i > d$ for $i \leq n-d$ and $p_i \leq d$ for $i > n-d$.

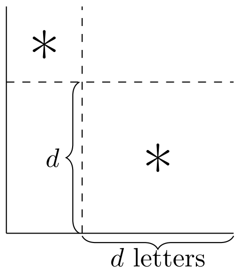


Figure 3.6: Plot of a minus-decomposable permutation, where $1 \leq d < k$.

In other words, for any $\pi \in \mathcal{C}$, to decide whether $\pi \in \mathcal{D}_M(\mathcal{C})$, it suffices to check whether there is an integer $d < k$, such that $E(\pi)$ consists of $n - d$ integers that are greater than d followed by d integers that are smaller or equal to d. This leads to the following languages:

$$L_d = \big\{\{d+1, \ldots, k\}^+ \{1, \ldots, d\}^d\big\} = \big\{\{d+1, \ldots, k\}\{d+1, \ldots, k\}^* \{1, \ldots, d\}^d\big\},$$

where L_d is a superset of sequences that are of similar form to the words in $E(\mathcal{D}_M(\mathcal{C}))$, with $d \in \{1, \ldots, k-1\}$ fixed.

Next, we will merge all possibilities of L_d,

$$\mathscr{L} = \bigcup_{d=1}^{k-1} L_d,$$

$\mathscr{L}$ contains all words representing k-bounded minus-decomposable permutations. Clearly $\mathscr{L}$ is regular.

Then from the above we can find that

$$E(\mathcal{D}_M(\mathcal{C})) = \mathscr{L} \cap E(\mathcal{C}),$$

which is a regular language, as by assumption $\mathcal{C}$ is a regular class with the language $E(\mathcal{C})$ and $\mathscr{L}$ is regular by construction. Thus the language $E(\mathcal{D}_M(\mathcal{C}))$ of minus-decomposable rank encoded permutations of a regular class is regular. □

Corollary 27. *Let $\mathcal{C}$ be a regular class. Then $\mathcal{I}_M(\mathcal{C})$, the set of all minus-indecomposable permutations of $\mathcal{C}$, is also regular under the rank encoding.*

Proof. Similar to the proof of Corollary 25 we obtain the above result by complementation.

Let $\mathcal{D}_M(\mathcal{C}) \subseteq \mathcal{C}$ be the regular set containing the minus-decomposable permutations. Then

$$\begin{aligned} &\mathcal{I}_M(\mathcal{C}) \cup \mathcal{D}_M(\mathcal{C}) = \mathcal{C} \;\Rightarrow\; E(\mathcal{I}_M(\mathcal{C})) \cup E(\mathcal{D}_M(\mathcal{C})) = E(\mathcal{C}) \\ &\Rightarrow\; E(\mathcal{I}_M(\mathcal{C})) = E(\mathcal{D}_M(\mathcal{C}))^C \cap E(\mathcal{C}), \end{aligned}$$

where $E(\mathcal{I}_M(\mathcal{C}))$ is the language of minus-indecomposable rank encoded permutations, and it is regular as $E(\mathcal{D}_M(\mathcal{C}))$ and $E(\mathcal{C})$ are regular. □

In conclusion, we have shown that the subset of minus-decomposable and minus-indecomposable permutations of a regular class are regular languages under the rank encoding. It is interesting to see that the result is the same as for plus-(in)decomposability but the approach to constructing the languages is different. Having shown the regularity of what is in effect the direct sum and skew sum of permutations, we will now investigate the languages of the direct and skew sums of classes.

3.4 Regularity of the Direct Sum and Skew Sum of Classes

Definitions. *direct sum of classes, skew sum of classes*

Let us start with recalling the definitions of the *direct sum* of two pattern classes $\mathcal{C}$ and $\mathcal{D}$,

$$\mathcal{C} \oplus \mathcal{D} = \{\rho : \rho = \pi \oplus \tau,\ \pi \in \mathcal{C},\ \tau \in \mathcal{D}\} = \{\rho : \rho = 12[\pi, \tau],\ \pi \in \mathcal{C},\ \tau \in \mathcal{D}\}$$

and the *skew sum* of two pattern classes,

$$\mathcal{C} \ominus \mathcal{D} = \{\rho : \rho = \pi \ominus \tau,\ \pi \in \mathcal{C},\ \tau \in \mathcal{D}\} = \{\rho : \rho = 21[\pi, \tau],\ \pi \in \mathcal{C},\ \tau \in \mathcal{D}\}.$$

Compared to the previous sections we do not assume the uniqueness of the plus- and minus-decompositions of $\rho \in \mathcal{C} \oplus \mathcal{D}$ or $\rho \in \mathcal{C} \ominus \mathcal{D}$, as we are taking any permutation of $\mathcal{C}$ in the decomposition of ρ.

First let us talk about the skew sum of classes and their behaviour under the rank encoding.

Theorem 28. *Let $\mathcal{C}$ and $\mathcal{D}$ be two regular classes under the rank encoding. Then the skew sum $\mathcal{E} = \mathcal{C} \ominus \mathcal{D}$ is a regular class under the rank encoding if and only if $\mathcal{D}$ is finite.*

Proof. Let $E(\mathcal{C}) \subseteq E(\Omega_k)$ and $E(\mathcal{D}) \subseteq E(\Omega_l)$ for some $k, l \in \mathbb{N}$, be regular classes under the rank encoding and $\mathcal{D}$ is infinite.

Assume that $E(\mathcal{E}) = E(\mathcal{C} \ominus \mathcal{D})$ is also regular. Let $\rho \in \mathcal{E}$, such that $\rho = \pi \ominus \tau = 21[\pi, \tau]$, where $\pi \in \mathcal{C}$ and $\tau \in \mathcal{D}$.

So any permutation $\rho \in \mathcal{E}$ has the form as shown in the figure below.

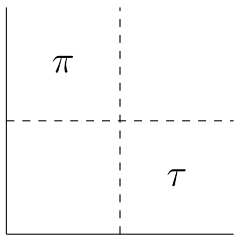


Figure 3.7: Plot of $\rho \in \mathcal{C} \ominus \mathcal{D}$.

Clearly encoding of ρ is

$$E(\rho) = (|\tau| + E(\pi))E(\tau).$$

As the language of the subwords corresponding to the permutations of $\mathcal{C}$ is shifted by the length of the permutations of $\mathcal{D}$, the alphabet of $\mathcal{E}$ is infinite, as $\mathcal{D}$ is infinite. Thus $E(\mathcal{E})$ is not regular, which contradicts our assumption.

Now let us assume that $E(\mathcal{C}) \subseteq E(\Omega_k)$ and $E(\mathcal{D}) \subseteq E(\Omega_l)$ for some $k, l \in \mathbb{N}$, are regular classes under the rank encoding and $\mathcal{D}$ is finite.

Then $\rho \in \mathcal{E} = \mathcal{C} \ominus \mathcal{D}$ has the form $\rho = 21[\pi, \tau]$ where $\pi \in \mathcal{C}$ and $\tau \in \mathcal{D}$. The encoding of any $\rho \in \mathcal{E}$ is $E(\rho) = (|\tau| + E(\pi))E(\pi)$ and as $\mathcal{D}$ is finite, the alphabet of $E(\mathcal{E})$ is finite and the language is

$$E(\mathcal{E}) = \bigcup_{i=\min(|\tau|, \tau \in \mathcal{D})}^{\max(|\tau|, \tau \in \mathcal{D})} (i + E(\mathcal{C}))E(\mathcal{D})^{[i]}.$$

The language $E(\mathcal{D})^{[i]}$ consists of all words of $E(\mathcal{D})$ of exactly length i, this is a finite language and thus regular. The union is a finite union of regular languages, so $E(\mathcal{E})$ is a regular language. □

On the other hand, we have the direct sum of any regular pattern classes which is a regular language.

Theorem 29. *Let $\mathcal{C}$ and $\mathcal{D}$ be two regular classes under the rank encoding. Then the direct sum $\mathcal{E} = \mathcal{C} \oplus \mathcal{D}$ is a regular class under the rank encoding.*

Proof. Let $E(\mathcal{C}) \subseteq E(\Omega_k)$ and $E(\mathcal{D}) \subseteq E(\Omega_l)$ for some $k, l \in \mathbb{N}$, be regular classes under the rank encoding. Further let $\mathcal{E} = \mathcal{C} \oplus \mathcal{D}$, then any permutation $\rho \in \mathcal{E}$ is $\rho = \pi \oplus \tau = 12[\pi, \tau]$, where $\pi \in \mathcal{C}$ and $\tau \in \mathcal{D}$. So $E(\rho) = E(\pi \oplus \tau) = E(12[\pi, \tau])$.

In lemma 23 we note that any plus-decomposition $\alpha = 12[\beta_1, \beta_2]$ in its rank encoding has the following form $E(\alpha) = E(\beta_1)E(\beta_2)$.

So for any $\rho \in \mathcal{E}$ we have $E(\rho) = E(\pi)E(\tau)$ where $\pi \in \mathcal{C}$ and $\tau \in \mathcal{D}$. Thus the whole language of the class $\mathcal{E}$ is

$$E(\mathcal{E}) = E(\mathcal{C})E(\mathcal{D}),$$

which is regular as $E(\mathcal{C})$ and $E(\mathcal{D})$ are regular languages and regularity is preserved under concatenation. $\square$

Overall we have shown that the skew sum of two regular classes is only regular if the class in the second summand is finite, whereas we have managed to show that for any two regular classes the direct sum of them is always regular.

A short note on separable classes. It is not always possible to find a regular language of a separable class under the rank encoding, unless the class is finite, as the generating function of separable classes has been found to be non-rational (proposition 1.4 of [AAV11]) but the generating function of regular languages is algebraic [CS59].

3.5 Decomposition by a Specific Simple Permutation

In this section we will only consider plus- and minus-indecomposable (*pmi*) permutations. We have shown that the sets of permutations with the properties of the two special cases of proposition 21 are regular languages under the rank encoding. We are now interested to see whether the set of permutations that are uniquely decomposable by a simple permutation is also a regular language under the rank encoding. In fact the set of permutations which in their unique block-decomposition have the same simple permutation is a regular language under the rank encoding. For that we will first look at the permutations of Ω_k that are decomposable by 2413 and then we will prove the case for any fixed simple permutation σ.

Theorem 30. *Let $E(\Omega_k)$ be the regular language of the rank encoded permutations with rank at most k. Then $\mathcal{D}_{2413} \subseteq \Omega_k$ the set of permutations in Ω_k having the block-decomposition $2413[\alpha_1, \alpha_2, \alpha_3, \alpha_4]$ with $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ non-empty is also regular under the rank encoding.*

Proof. Let $E(\Omega_k)$ with $k \in \mathbb{N}$ and $E(\mathcal{D}_{2413})$ be the set of rank encoded permutations of $\mathcal{D}_{2413}$.

Let $\pi \in \mathcal{D}_{2413}$ then $\pi = 2413[\alpha_1, \alpha_2, \alpha_3, \alpha_4]$, where $|\pi| = n$, $|\alpha_1| = a$, $|\alpha_2| = b$, $|\alpha_3| = c$ and $|\alpha_4| = d$.

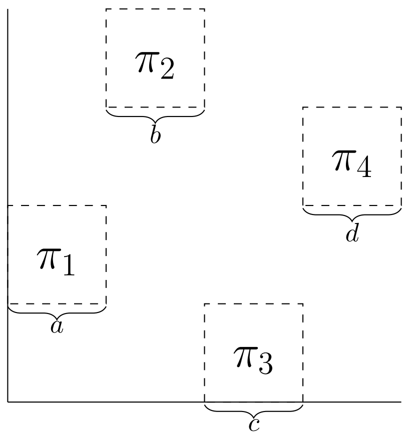


Figure 3.8: Plot of π, with the lengths of the intervals indicated.

We can see that $E(\pi)$ will end as follows

$$E(\pi) = E(\pi)[1, \dots, n - c - d]\ E(\alpha_3)\ E(\alpha_4).$$

This is due to the fact that the relative position of α_3 with respect to 2413 is in the lowest interval, so there is nothing to the right and below the interval π_3. Similarly for π_4. Further notice that $n - c - d = a + b$.

The first part of $E(\pi)$, which corresponds to π_1, is $E(\pi)[1, \ldots, a] = c + E(\alpha_1)$ as all points of π_3 lie below and to the right of all points of π_1.

Similarly the next part of $E(\pi)$, which corresponds to π_2, is $E(\pi)[a+1, \ldots, a+b] = (c+d)+E(\alpha_2)$ as all points in π_2 lie above and to the left of π_3 and π_4.

Thus

$$E(\pi) = \big(c + E(\alpha_1)\big)\ \big((c+d) + E(\alpha_2)\big)\ E(\alpha_3)\ E(\alpha_4).$$

Note that as $\mathcal{D}_{2413} \subseteq \Omega_k$ and $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are non-empty, we have a maximal rank and we can let the language be the following:

$$L_{c,d} = \big((c + E(\Omega_{k-c})) \setminus \{\varepsilon\}\big)\ \Big(((c+d) + E(\Omega_{k-(c+d)})) \setminus \{\varepsilon\}\Big)\ (E(\Omega_k)^{(c)} \setminus \{\varepsilon\})\ (E(\Omega_k)^{(d)} \setminus \{\varepsilon\}),$$

where $1 \leq c \leq k-2$, $1 \leq d \leq k-2$ and $2 \leq c+d \leq k-1$, as the values of the part of $E(\pi)$ representing α_2 must be strictly larger than $c+d$ and less than k. Recall that $E(\Omega_k)^{(c)}$ is the language of words of $E(\Omega_k)$ up to and including length c.

By concatenation and lemma 22, $L_{c,d}$ is regular.

Now we want to build a language allowing for all possible lengths of permutations α_3, α_4,

$$\mathscr{L} = \bigcup_{c=1}^{k-2} \bigcup_{d=1}^{k-1-c} L_{c,d},$$

which is regular, as regularity is preserved under finite union. As $\mathscr{L}$ is an abstraction of the words corresponding to 2413-decomposable permutations

$$E(\mathcal{D}_{2413}) \subseteq \mathscr{L} \cap E(\Omega_k)$$

is also regular as $\mathscr{L}$ and $E(\Omega_k)$ are regular and regularity is preserved under intersection.

Now it is enough to prove that $E(\mathcal{D}_{2413}) \supseteq \mathscr{L} \cap E(\Omega_k)$. Let $w \in \mathscr{L} \cap E(\Omega_k)$, there exists $\pi \in \Omega_k$ such that $w = E(\pi)$.

Then there exist some non-zero c and d such that $E(\pi) \in L_{c,d}$. So $E(\pi)$ has the form $E(\pi) = w_1 w_2 w_3 w_4$ where,

$$\begin{aligned}
&w_1 \in (c + E(\Omega_{k-c})) \setminus \{\varepsilon\} \\
&w_2 \in \big((c+d) + E(\Omega_{k-(c+d)})\big) \setminus \{\varepsilon\} \\
&w_3 \in E(\Omega_k)^{(c)} \setminus \{\varepsilon\} \\
&w_4 \in E(\Omega_k)^{(d)} \setminus \{\varepsilon\}.
\end{aligned}$$

From this decomposition we know that there exist some permutations $\alpha_1, \dots, \alpha_4$ such that

$$\begin{aligned} w_1 &= c + E(\alpha_1) \\ w_2 &= (c+d) + E(\alpha_2) \\ w_3 &= E(\alpha_3) \\ w_4 &= E(\alpha_4), \end{aligned}$$

$|\alpha_3| = c,\ |\alpha_4| = d.$

Since $\alpha_1, \dots, \alpha_4$ are subpermutations of π, $\alpha_1, \dots, \alpha_4 \in \Omega_k$.

Let $\tau = 2413[\alpha_1, \dots, \alpha_4]$. Then clearly $\tau \in \mathcal{D}_{2413}$. Furthermore, $E(\tau) = w_1w_2w_3w_4 = E(\pi)$ thus by the definition of the rank encoding $\tau = \pi$ and so $\pi \in \mathcal{D}_{2413}$. Which implies that

$$E(\mathcal{D}_{2413}) = \mathscr{L} \cap E(\Omega_k).$$

□

Let us now look at the general case where the language of rank encoded permutations is the inflation over the same simple permutation σ, in other words permutations with the same simple permutation σ in their block-decomposition.

Definitions. *maximal interval*

For the simple σ in the decomposition of a pmi permutation $\pi = \sigma[\alpha_1, \dots, \alpha_n]$ to be found, we will have to find the maximal proper intervals of length $< |\pi|$ in the permutation π that is being decomposed. An interval in a permutation is said to be *maximal* if it is maximal under the partial order of set inclusion.

Lemma 31. *The maximal proper intervals in a pmi permutation are all disjoint.*

Proof. Let π be a pmi permutation, and α, β be maximal proper intervals such that $\alpha \cap \beta \neq \emptyset$.

Then if $\alpha \cup \beta \neq \pi$, by definition of maximal intervals there is a maximal interval $\gamma = \alpha \cup \beta$, $\gamma \neq \pi$ and α, β are not maximal.

If $\alpha \cup \beta = \pi$ then π is plus- or minus-decomposable with the permutations in the decomposition being $\alpha \setminus \beta$ and β. We can see that $\alpha \setminus \beta$ and $\alpha \cap \beta$ are an interval itself, as both α and β are intervals, so there cannot be any points in the plot of π that lie directly above, below, to the left or right of α or β, see figure 3.9.

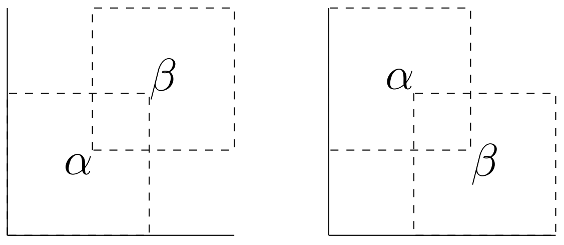


Figure 3.9: Possible positioning of maximal intervals α, β in π with $\alpha \cap \beta \neq \emptyset$, $\alpha \cup \beta = \pi$.

We can see from those figures that π will not be pmi, which is a contradiction with our assumption. □

Theorem 32. *If $\pi = \sigma[\alpha_1, \dots, \alpha_n]$ is a pmi permutation, then in its unique block-decomposition with σ simple, the α_i are all order isomorphic to the maximal proper intervals of π.*

Proof. Let $\pi = \sigma[\alpha_1, \dots, \alpha_n]$, with σ simple and α_i, for some $1 \leq i < n$ corresponding to an interval a_i that is not maximal in π. So $a_i \subsetneq b_i$, where b_i is maximal. Any a_j which meets b_i is contained in it, by lemma 31. So b_i is a union of consecutive a_j's.

Thus there are p, q with $p \leq i \leq q$, $p \leq q - 1$,

$$b_i = \bigcup_{j=p}^{q} a_j.$$

So $\sigma(p) \dots \sigma(q)$ forms an interval in σ. This contradicts with the assumption that σ is simple.

Thus, in a pmi permutation $\pi = \sigma[\alpha_1, \dots, \alpha_n]$ the α_i are order isomorphic to maximal proper intervals in π. □

Theorem 33. *Let $E(\Omega_k)$ be the regular language of the rank encoded permutations with rank at most k. Let $|\sigma| > 2$ be a simple permutation, then the set $\mathcal{D}_\sigma \subseteq \Omega_k$ of σ-decomposable permutations of Ω_k is also regular under the rank encoding.*

Proof. We know each block-decomposition with a specific simple permutation has a certain structure in its plot. We will be looking at recreating this structure in the language of all k rank encoded σ decomposable permutations. At the same time we will show the steps of the proof on an example, namely when $\sigma = 25314$ and α_1, α_2, α_3, α_4 and α_5 non-empty.

Let $|\sigma| = n$ and $\pi = \sigma[\alpha_1, \dots, \alpha_n] \in \mathcal{D}_\sigma \subseteq \Omega_k$, with arbitrary α_i non-empty. For notation purposes let

$$\xi_i = \sum_{\substack{j>i \\ \sigma(j)<\sigma(i)}} |\alpha_i|.$$

For $\sigma = 25314$ we have

$$\xi_1 = |\alpha_4|, \quad \xi_2 = |\alpha_3| + |\alpha_4| + |\alpha_5|, \quad \xi_3 = |\alpha_4|, \quad \xi_4 = \xi_5 = 0.$$

In π we can see that for any $i \in \{1, \dots, n\}$ and every letter in $E(\pi)[x, \dots, y]$, which is the subword of $E(\pi)$ corresponding to $\pi_i = \pi(x) \dots \pi(y)$, is strictly greater than ξ_i, as the ranks of the points in π_i are greater than the number of points to the right and below of π_i, in the plot of π.

Furthermore, for any value $\sigma(i)$ that is not a left-to-right maximum in σ, $|\alpha_i| < k$ because there exists an $j < i$ such that $\sigma(j) > \sigma(i)$. So for any $z \in \{u, \dots, v\}$, where $\pi_j = \pi(u) \dots \pi(v)$, $E(\pi)[z] > |\alpha_i|$ but $E(\pi)[z] \leq k$.

Recall $\pi_i = \pi(x)\pi(x + 1) \dots \pi(y)$ to be the interval of π corresponding to the block of α_i at position $\sigma(i)$ of the σ-decomposition of π. We divide the i of $\{1, \dots, n\}$ with respect to the indexing of σ into the following sets.

Max The set of i, where $\sigma(i)$ is a left-to-right maximum.

Min The set of i, where $\sigma(i)$ is a right-to-left minimum.

$\mathcal{R}$ The set of i, where $\sigma(i)$ is neither a left-to-right maximum nor a right-to-left minimum.

Note that $Max \cap Min = \emptyset$ because σ is simple and $|\sigma| \geq 3$, as otherwise σ is plus-decomposable. So for $\sigma = 25314$ the above sets are $Max = \{1, 2\}$, $Min = \{4, 5\}$, $\mathcal{R} = \{3\}$.

When $i \in Max$ the subword of $E(\pi)$ corresponding to the size unrestricted interval π_i will be contained in the languages

$$\xi_i + E(\Omega_{k-\xi_i}).$$

When $i \in Min$ the subword of $E(\pi)$ corresponding to the size restricted interval π_i will be contained in the languages

$$E(\Omega_k)^{(y-x+1)}.$$

When $i \in \mathcal{R}$ the subword of $E(\pi)$ corresponding to the size restricted interval π_i will be contained in the languages

$$(\xi_i + E(\Omega_{k-\xi_i}))^{(y-x+1)}.$$

Let $Max^c = Min \cup \mathcal{R}$ be the set of indices that correspond to the $\sigma(i)$ that are not left-to-right maxima. Additionally define the sequence $\mathcal{M}^c = \langle |\pi_i| : i \in Max^c \rangle$. In our example that means that $Max^c = \{3, 4, 5\}$ and the sequence is $\mathcal{M}^c = \langle |\alpha_3|, |\alpha_4|, |\alpha_5| \rangle$

We can now construct a regular language $L_{\mathcal{M}^c}$ which consists of a concatenation of the above regular languages based on the types of indices occurring in σ and is bound to a specific sequence $\mathcal{M}^c$. So for our example σ we have the language, for a specific sequence $\mathcal{M}^c = \langle |\alpha_3|, |\alpha_4|, |\alpha_5| \rangle$

$$L_{\mathcal{M}^c} = (\xi_1 + E(\Omega_{k-\xi_1}))(\xi_2 + E(\Omega_{k-\xi_2}))(\xi_3 + E(\Omega_k)^{(|\alpha_3|)})(E(\Omega_k)^{(|\alpha_4|)})(E(\Omega_k)^{(|\alpha_5|)})$$

Next the regular language $\mathscr{L}$ is the finite union over all sequences $\mathcal{M}^c$,

$$\mathscr{L} = \bigcup_{\mathcal{M}^c} L_{\mathcal{M}^c}.$$

For our specific $\sigma = 25314$ we have

$$\mathscr{L} = \bigcup_{|\alpha_3|} \bigcup_{|\alpha_4|} \bigcup_{|\alpha_5|} L_{\mathcal{M}^c}.$$

We have shown that the general regular language $\mathscr{L}$ contains the language $E(\mathcal{D}_\sigma)$,

$$E(\mathcal{D}_\sigma) \subseteq \mathscr{L} \cap E(\Omega_k).$$

We now need to show that there are no other rank encoded permutations within that language. So we are going to prove that $E(\mathcal{D}_\sigma) \supseteq \mathscr{L} \cap E(\Omega_k)$. Let $w \in \mathscr{L} \cap E(\Omega_k)$.

We know that for a specific sequence $\mathcal{M}^c$, $w \in L_{\mathcal{M}^c}$. This means that w consists of factors $w = w_1 \dots w_n$ where

$$\begin{aligned} w_i &\in \eta_i + E(\Omega_{k-\eta_i}) \text{ when } i \in Max \\ w_j &\in E(\Omega_k)^{(y_j - x_j + 1)} \text{ when } j \in Min \\ w_h &\in \eta_h + E(\Omega_{k-\eta_h})^{(y_h - x_h + 1)} \text{ when } h \in \mathcal{R}, \end{aligned}$$

where if $w_i = w(x) \dots w(y)$ and $w_j = w(z) \dots w(v)$ then

$$\eta_i = \sum_{\substack{v > z > y \\ \sigma(j) < \sigma(i)}} v - z + 1.$$

From this decomposition we know that there exists some $\alpha_1, \dots, \alpha_n$ such that

$$\begin{aligned} w_i &= \eta_i + E(\alpha_i) \text{ when } i \in Max \\ w_j &= E(\alpha_j) \text{ when } j \in Min \\ w_h &= \eta_h + E(\alpha_h) \text{ when } h \in \mathcal{R}, \end{aligned}$$

$|\alpha_j| = y_j - x_j + 1,\ |\alpha_h| = y_h - x_h + 1$, for $w_j = w(x_j) \dots w(y_j)$ and $w(h) = w(x_h) \dots w(y_h)$.

Since $\alpha_1, \dots, \alpha_n$ are subpermutations of permutations in Ω_k, $\alpha_1, \dots, \alpha_n \in \Omega_k$. Let $\tau = \sigma[\alpha_1, \dots, \alpha_n]$, then clearly $\tau \in \mathcal{D}_\sigma$ and $E(\tau) = w$ as $\xi_i = \eta_i$.

Thus

$$E(\mathcal{D}_\sigma) = \mathscr{L} \cap E(\Omega_k),$$

and $E(\mathcal{D}_\sigma)$ is regular. □

In conclusion, we have managed to show that the sets of σ-decomposable permutations of a regular class are also a regular language under the rank encoding. It is of interest to see whether this result for permutations can be extended onto the inflation of classes.

3.6 Regularity of the Inflation of Classes

Definitions. *inflation of classes*

Let us recall the definitions of the inflation of classes over permutations, while proving or disproving the existence of a regular language of the classes with these properties under the rank encoding. The *inflation* of classes $\mathcal{C}_i$ over a simple permutation σ of length n is the class $\mathcal{D}$,

$$\mathcal{D} = \sigma[\mathcal{C}_1, \dots, \mathcal{C}_n] = \{\sigma[\gamma_1, \dots, \gamma_n] : \gamma_i \in \mathcal{C}_i\}.$$

Lemma 34. *Let σ be any simple permutation of length n, $\sigma[\mathcal{C}_1, \dots, \mathcal{C}_n] = \mathcal{D}$ be an inflation of σ by the regular classes $\mathcal{C}_i \subseteq \Omega_k$. Then $E(\mathcal{D})$ is a regular language under the rank encoding if and only if $\mathcal{C}_j$ is finite when $\sigma(j)$ is not a left-to-right maximum.*

We will be using a similar way to construct the language as in the proof of theorem 33.

Proof. Let $\mathcal{D} = \sigma[\mathcal{C}_1, \dots, \mathcal{C}_n]$, with the $\mathcal{C}_i$ being regular classes and the $\mathcal{C}_j$ for $\sigma(j)$ not a left-to-right maximum being finite classes.

Then the languages $E(\mathcal{C}_i)$ when $\sigma(i)$ is a left-to-right maximum are at most shifted by the sum of the lengths of the longest permutations of the classes $\mathcal{C}_j$, $j > i$ and $\sigma(j) < \sigma(i)$. Further we split the set of indices of σ that are not left-to-right maxima into two sets, $Min = \{i : \sigma(i) \text{ is right-to-left minimum}\}$ and $\mathcal{R} = \{i : \sigma(i) \text{ is not a left-to-right maximum and } i \notin Min\}$. Then the languages $E(\mathcal{C}_i)$, $i \in \mathcal{R}$ are also being at most shifted by the sum of the lengths of the longest permutations of the classes $\mathcal{C}_j$, if $j > i$ and $\sigma(j) < \sigma(i)$. Lastly the languages $E(\mathcal{C}_i)$ when $i \in Min$ remain the same.

We now can concatenate these languages according to their position in the decomposition. As all the languages are regular and regularity is preserved under concatenation the language

$$\mathcal{D} = \sigma[\mathcal{C}_1, \dots, \mathcal{C}_n]$$

is regular.

Now let us assume that not all $\mathcal{C}_j$ of $\mathcal{D} = \sigma[\mathcal{C}_1, \ldots, \mathcal{C}_n]$, where $\sigma(j)$ is not a left-to-right maximum, are finite.

Then the languages that correspond to the $\mathcal{C}_i$, $\sigma(i)$ is a left-to-right maximum, will be shifted by the lengths of the permutations of $\mathcal{C}_j$. But as $\mathcal{C}_j$ is infinite, the alphabet of that shifted language is also going to be infinite. Thus the language will not be regular and so $E(\mathcal{D})$ is not regular if not all $\mathcal{C}_j$ of $\mathcal{D} = \sigma[\mathcal{C}_1, \ldots, \mathcal{C}_n]$ where $\sigma(j)$ is not a left-to-right maximum, are finite. □

So we have shown that the inflation of regular classes over a fixed permutation σ is regular if the classes $\mathcal{C}_i$ corresponding to the $\sigma(i)$ which are non-left-to-right maximum elements in σ are finite.

3.7 Language of Simple Permutations

As seen in [AA05], knowing the set of simple permutations is highly useful for wreath closed classes. This is only one application of simple permutations in classes as we will show in the following sections.

Definitions. *gap sizes*

We have to introduce a few more concepts before being able to prove our next theorem, which looks at the language of simple permutations under the rank encoding. Given a finite subset $A \subset \mathbb{N}$, the *gap sizes* of A, $gs(A)$, are defined as follows. Let

$$\{1, \ldots, \max(A)\} \setminus A = \{b_1, \ldots, e_1\} \cup \ldots \cup \{b_y, \ldots e_y\},$$

where $b_i \leq e_i < b_{i+1} - 1$ for $i \in \{1, \ldots, y\}$, then $gs(A)$ is the sequence $\langle e_1 - b_1 + 1, \ldots, e_y - b_y + 1\rangle$, for $y \in \mathbb{N}$. For example, $gs(\{1, 4, 6\}) = \langle 2, 1\rangle$. We apply this notion to prefixes of permutations by considering them as sets of values.

Example. *In figure 3.10 we look at the prefix* 45 *of a permutation, at that point we have the gap sizes* $\langle 3\rangle$. *When then a new maximal element* 8 *is added, the gap sizes are* $\langle 3, 2\rangle$. *Further when* 1 *is added the gaps sizes change to* $\langle 2, 2\rangle$.

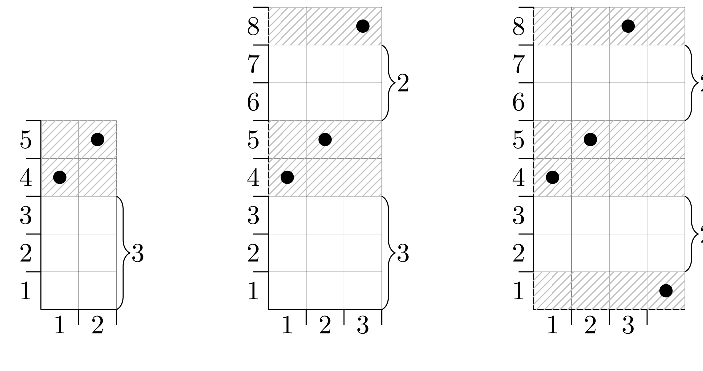


Figure 3.10: Plots of parts 45, 458 and 4581 with the gaps indicated.

We use $\sum gs(A)$ in the natural way to denote the sum of the sequence $gs(A)$.

Observe that for a permutation with maximal rank k, the sum $\sum gs(\pi(1)\dots\pi(y)) \leq k-1$ for any prefix of π. If the sum is $> k-1$ then the maximal element of the prefix $\pi(1)\dots\pi(y)$ has rank $> k$, in the encoding of π.

Furthermore we will use the notation $gs'(w)$ where $w = E(\pi)[1,\dots,y]$, $y \leq |\pi|$. Clearly, $gs'(w) = gs(\pi(1)\dots\pi(y))$.

Definitions. *gap automaton*

Based on gap sizes we can now introduce the *gap automaton* construct, $\mathcal{G}(k, A) = (\Sigma, S, \delta, s_1, A)$. The alphabet of $\mathcal{G}(k, A)$ is $\Sigma = \{1,\dots,k\}$, $k \in \mathbb{N}$, the set of states S is the set of all possible gap sizes for rank k, so all possible gap sizes that sum up to at most $k-1$, a transition function δ, the start state s_1 being the empty gap size, and the set of accept states A will vary to give different automata for different applications. We define the transition function as the following pseudo-code algorithm. This algorithm takes any gap sizes of a prefix w of a permutation and finds the next gap sizes when a letter r is appended to w.

Algorithm 1 Calculate the next gap sizes/state of the gap automaton $\mathcal{G}(k, A)$.

```
Input: A state in form of gap sizes $gs'(w) = \langle g_1,\dots,g_x \rangle$ and a letter $r$
 1: if $r \leq \sum gs'(w)$ then
 2:     Find the least $i \in \{1,\dots,x\}$ such that $r \leq g_1 + \cdots + g_i$
 3:     if $r = g_1 + \cdots + g_i$ then
 4:         $h \leftarrow gs'(w)$
 5:         $h_i \leftarrow g_i - 1$
 6:     else                                                  ▷ $r < g_1 + \cdots + g_i$
 7:         $h \leftarrow gs'(w)$
 8:         Insert new element $r - (g_1 + \cdots + g_{i-1}) - 1$ to $h$ at position $i$
 9:         $h_{i+1} \leftarrow g_i - h_{i-1} - 1$
10:     end if
11: else
12:     $h \leftarrow gs'(w)$
13:     $h_{x+1} \leftarrow r - \sum gs'(w) - 1$
14: end if
15: if $0 \in h$ and $|h| > 1$ then
16:     Remove all 0's from $h$
17: end if
    return Gap sizes $h = gs'(wr)$
```

It is easy to see that the language accepted by the automaton $\mathcal{G}(k, \{\langle\emptyset\rangle, \langle 0\rangle\})$ is equal to $E(\Omega_k)$, $k \in \mathbb{N}$.

Example. *The gap automaton $\mathcal{G}(3, \emptyset)$ is depicted in figure 3.11.*

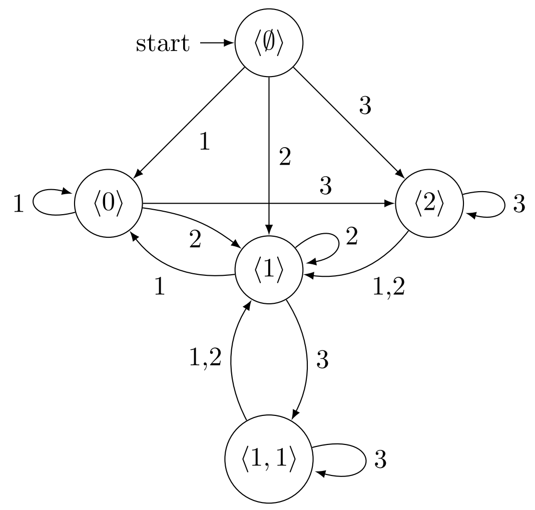


Figure 3.11: Gap automaton with $k = 3$.

Theorem 35. *The set of all non-simple permutations $\mathcal{NS}_k$ of Ω_k is regular under the rank encoding.*

Proof. Let π be a non-simple permutation of length n. Then π will contain at least one non-trivial interval. Let $\mathbf{I} = \pi(r)\pi(r+1)\ldots\pi(r+s)$, $1 \le r < r+s \le n$, $s < n-1$, be such an interval with r minimal.

In general the plot of π will be as in figure 3.12

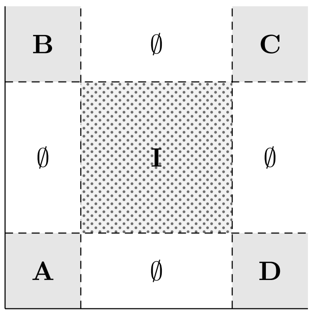


Figure 3.12: Plot of position of interval **I** in a non-simple permutation.

where

$$
\begin{aligned}
\mathbf{A} =& \{(x, y)\ :\ \pi(x) = y,\ x < r,\ y < \min(I)\} \\
\mathbf{B} =& \{(x, y)\ :\ \pi(x) = y,\ x < r,\ y > \max(I)\} \\
\mathbf{C} =& \{(x, y)\ :\ \pi(x) = y,\ x > r+s,\ y > \max(I)\} \\
\mathbf{D} =& \{(x, y)\ :\ \pi(x) = y,\ x > r+s,\ y < \min(I)\}\ .
\end{aligned}
$$

We will introduce the following abbreviations:

$$\begin{aligned}
\mathscr{L}_1 &= \bigcup_{l=1}^{k-1} \mathscr{P}_l \bigcup_{m=l}^{k-1} (m + E(\hat{\Omega}_{k-m}))\Sigma^* \\
\mathscr{L}_2 &= \bigcup_{j=1}^{k-1} (j + E(\hat{\Omega}_{k-j}))\Sigma^* \\
\mathscr{L}_3 &= \bigcup_{a=2}^{k-1} \bigcup_{b=0}^{k-1-a} \mathscr{Q}_{a,b} \bigcup_{i=0}^{a-2} (((b+i) + E(\hat{\Omega}_{k-(b+i)})))^{(a-i)}\Sigma^* \\
\mathscr{L}_4 &= E(\Omega_k \setminus \{\varepsilon\})E(\Omega_k \setminus \{\varepsilon\})\Sigma^*
\end{aligned}$$

where

Σ is the alphabet $\{1, \ldots, k\}$, $k \in \mathbb{N}$, $k \geq 3$.

$\mathscr{P}_l$ is the language of prefixes of k rank encoded permutations, where $\sum gs'(w) = l$. From figure 3.10 we can see that $\mathscr{P}_3$ would contain $E(45) = 44$, $\mathscr{P}_5$ would contain $E(458) = 446$ and $\mathscr{P}_4$ would contain $E(4581) = 4461$.

$\mathscr{Q}_{i,j}$ is the language of prefixes of k rank encoded permutations, where in $gs'(w) = \langle g_1, \ldots, g_n \rangle$ there is a gap g_x, $1 \leq x \leq n$, of size i and sum of the gaps $g_1, \ldots g_{x-1}$ equals to j. From figure 3.10 for example $E(45) = 44$ lies in $\mathscr{Q}_{3,0}$, $E(458) = 446$ will lie in $\mathscr{Q}_{3,0}$ and $\mathscr{Q}_{2,3}$, and $E(4581) = 4461$ will lie in $\mathscr{Q}_{2,0}$ and $\mathscr{Q}_{2,2}$.

$i + E(\Omega_{k-i})$ is the language of $E(\Omega_{k-i})$, $i \in \mathbb{N}$, with the alphabet shifted upwards by i.

$E(\Omega_k)^{(i)}$ is the sublanguage of $E(\Omega_k)$ containing the words of length $\leq i$, $i \in \mathbb{N}$.

$E(\hat{\Omega}_k)$ is the sublanguage of $E(\Omega_k)$ containing the words of length > 1.

Before showing that the encoding of any non-simple permutation will be found in one of the above languages, we will prove that each is a regular language.

It is clear that Σ^*, $i + E(\Omega_{k-i})$, $E(\Omega_k)^{(i)}$, $E(\hat{\Omega}_k)$ and $E(\Omega_k \setminus \{\varepsilon\})$ are regular languages.

Both $\mathscr{P}_l$ and $\mathscr{Q}_{a,b}$ are recognised by gap automaton $\mathcal{G}(k, A)$, A for $\mathscr{P}_l$ contains the states where the sum of the gap sizes $gs'(w)$, $\sum gs'(w) = l$. The set of final states A of $\mathscr{Q}_{a,b}$ contains the states with gap sizes $gs'(w) = \langle g_1, \ldots, g_n \rangle$ which contain a gap $g_x = a$ and the sum $\sum_{i=1}^{x-1} g_i = b$.

Clearly $\mathcal{G}(k, A)$ is a well-defined and finite automaton, and with the final states the automata of $\mathscr{P}_l$ and $\mathscr{Q}_{a,b}$ define regular languages which are easily checked to be non-empty.

$\mathscr{L}_1, \mathscr{L}_2, \mathscr{L}_3$ and $\mathscr{L}_4$ are all concatenations and finite unions of regular languages. Thus all four languages are regular.

Now we will show that the encodings of $E(\pi)$ for π non-simple lie in these languages. We will look at what languages $E(\pi)$ lies in depending on the positioning of the interval $\mathbf{I}$.

As the interval $\mathbf{I}$ is non-trivial, not all $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ can be empty simultaneously.

If $\mathbf{B} = \emptyset$ and $\mathbf{A}, \mathbf{D} \neq \emptyset$ then $E(\pi) \in \mathscr{L}_1$, as the points and gaps of $\mathbf{A}$ corresponding to the prefix of $E(\pi)$ lie in $\mathscr{P}_l$, $l \leq |\mathbf{D}|$, which is then followed by the part of $E(\pi)$ that is the interval $E(\mathbf{I})$ which lies in $m + E(\hat{\Omega}_{k-m})$ and the points of $\mathbf{C}$ and $\mathbf{D}$ lie in Σ^*.

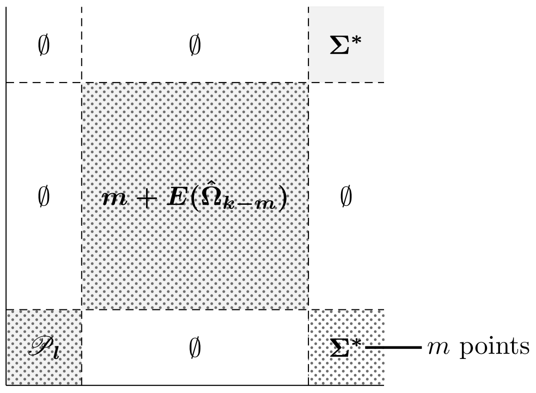


Figure 3.13: Plot of permutations represented by rank encodings in $\mathscr{L}_1$.

If $\mathbf{A}, \mathbf{B} = \emptyset$ and $\mathbf{D} \neq \emptyset$ then $E(\pi) \in \mathscr{L}_2$, because there are no points preceding the interval which lies in $j + E(\hat{\Omega}_{k-j})$ for $j = |\mathbf{D}|$, which is the shifted language of $E(\Omega_k)$ and there are j points in $\mathbf{D}$.

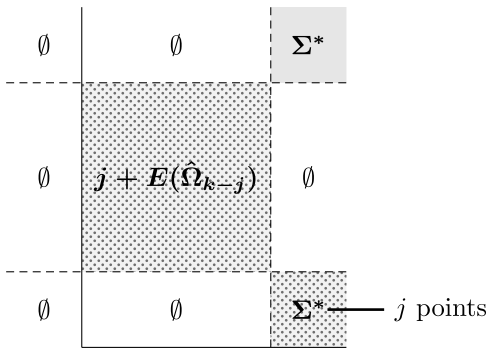


Figure 3.14: Plot of permutations represented by rank encodings in $\mathscr{L}_2$.

If $\mathbf{B} \neq \emptyset$ then $E(\pi) \in \mathscr{L}_3$, as the rank encoded points of $\mathbf{A} \cup \mathbf{B}$ will be in $\mathscr{Q}_{a,b}$ for $a = |\mathbf{I}| + i$, $b = |\mathbf{D}| - i$, and if there are no points in $\mathbf{A}$ then $b = 0$ otherwise the points of $\mathbf{A}$ lie also in $\mathscr{Q}_{a,b}$, the part of the word representing the interval will be $((b + i) + E(\hat{\Omega}_{k-(b+i)}))^{(a-i)}$, and the points of $\mathbf{C}$ and $\mathbf{D}$ lie in Σ^*.

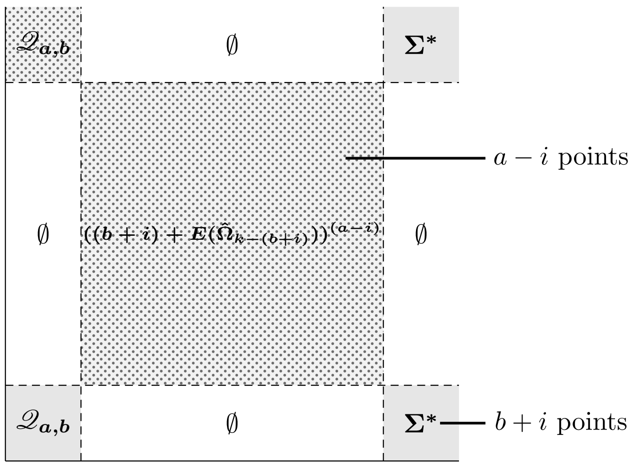


Figure 3.15: Plot of permutations represented by rank encodings in $\mathscr{L}_3$.

Finally if $\mathbf{B}, \mathbf{D} = \emptyset$ then because r is minimal $\mathbf{A}$ must also be empty and so $E(\pi) \in \mathscr{L}_4$, because $E(\mathbf{I})$ lies in the first $E(\Omega_k \setminus \{\varepsilon\})$ and the points of $\mathbf{C}$ lie in $E(\Omega_k \setminus \{\varepsilon\})\Sigma^*$. Incidentally $\mathscr{L}_4$ includes the rank encodings of plus-decomposable permutations.

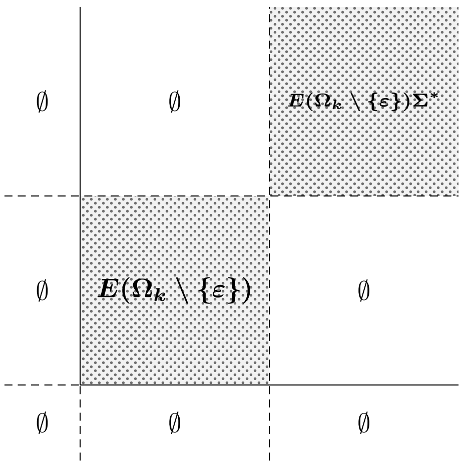

$\emptyset$
$\emptyset$
$E(\Omega_k \setminus \{\varepsilon\})\Sigma^*$
$\emptyset$
$E(\Omega_k \setminus \{\varepsilon\})$
$\emptyset$
$\emptyset$
$\emptyset$
$\emptyset$

Figure 3.16: Plot of permutations represented by rank encodings in $\mathscr{L}_4$.

All the above cases of the placement of the interval $\mathbf{I}$ have shown that the union of all four languages $\mathscr{L}_1, \mathscr{L}_2, \mathscr{L}_3$ and $\mathscr{L}_4$ does indeed include all rank encoded non-simple permutations, so

$$E(\mathcal{NS}_k) \subseteq (\mathscr{L}_1 \cup \mathscr{L}_2 \cup \mathscr{L}_3 \cup \mathscr{L}_4) \cap E_{\Omega_k}.$$

We also have to prove that this language excludes all words corresponding to simple permutations under the rank encoding. Let π be simple and assume $E(\pi) \in E(\mathcal{NS}_k)$.

- If $E(\pi) \in \mathscr{L}_1$, then $m + E(\hat{\Omega}_{k-m})$ will represent a subword of $E(\pi)$ that corresponds to a non-trivial interval in π, as $E(\hat{\Omega}_k)$ contains words of length > 1. Further, as $m + E(\hat{\Omega}_{k-m})$ is a shifted language of valid rank encodings, we will have no points in the plot of $E(\pi)$ strictly to the right of the parts of π corresponding to the words in $m + E(\hat{\Omega}_{k-m})$ as this part is a non-trivial interval.

- If $E(\pi) \in \mathscr{L}_2$, then $j + E(\hat{\Omega}_{k-j})$ will represent a subword of $E(\pi)$ that corresponds to a non-trivial interval in π.
- If $E(\pi) \in \mathscr{L}_3$, then $((b+i) + E(\hat{\Omega}_{k-(b+i)}))^{(a-i)}$ will represent a subword of $E(\pi)$ that corresponds to a non-trivial interval in π.
- Finally if $E(\pi) \in \mathscr{L}_4$ then either $E(\Omega_k \backslash \{\varepsilon\})$ will represent a subword of $E(\pi)$ that corresponds to a non-trivial interval in π.

We have a contradiction, so $E(\pi) \notin E(\mathcal{NS}_k)$. Thus $E(\mathcal{NS}_k)$ is the language of all rank encoded non-simple permutations of Ω_k.

So we have proven that indeed the set of non-simple permutations $\mathcal{NS}_k$ of Ω_k is a regular language under the rank encoding. □

Corollary 36. *The set of simple permutations $\mathcal{S}_k$ of Ω_k is regular under the rank encoding.*

Proof. Let $\mathcal{NS}_k$ be the set of all non-simple permutations of Ω_k. As described and proved above $E(\mathcal{NS}_k)$ is regular. Then the set of simple permutations $\mathcal{S}_k$ is

$$\mathcal{S}_k = \Omega_k \setminus \mathcal{NS}_k.$$

Thus, the language of simple permutations of Ω_k under the rank encoding is

$$E(\mathcal{S}_k) = E(\Omega_k \setminus \mathcal{NS}_k) = E(\Omega_k) \cap E(\mathcal{NS}_k)^C.$$

As regularity is preserved under intersection and complement, $E(\mathcal{S}_k)$, the set of all simple permutations of Ω_k, is regular. □

Definitions. *exceptional permutation*

In [PR12] Pierrot and Rossin discuss the chains of simple permutations that are created through one or two point deletions of elements. If a point is removed from most simple permutations the result will be another simple permutation. But there is a type of simple permutations that requires that two points are removed to result in a simple permutation. These permutations are called *exceptional permutations* [AA05, Bri10, PR12]. A simple permutation is exceptional if it is one of the following types

1. $246\ldots(2n)135\ldots(2n-1)$
2. $(2n-1)(2n-3)\ldots31(2n)(2n-2)\ldots42$
3. $(n+1)1(n+2)2(n+3)3\ldots(2n)n$
4. $n(2n)(n-1)(2n-1)(n-2)(2n-2)\ldots1(n+1)$,

where $n \in \mathbb{N}$.

Example. *There are four exceptional permutations of length 8*

24681357

75318642

51627384

48372615

and their plots are shown in figure 3.17.

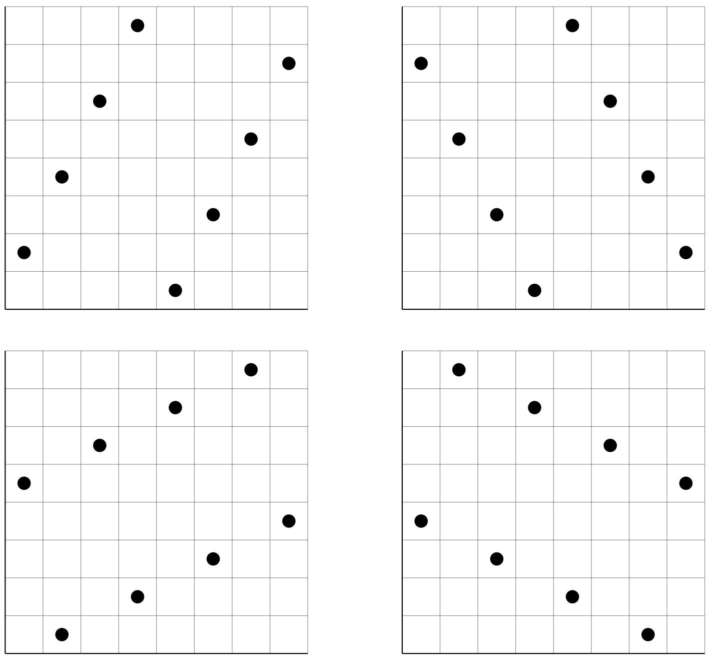


Figure 3.17: Plots of the four different types of exceptional permutations of length 8.

Lemma 37. *The set of exceptional permutations of Ω_k is finite.*

Proof. We will show that for each type of exceptional permutation there are finitely many permutations in Ω_k using their language under the rank encoding.

The exceptional permutations of type (1) have the form

$$246\dots(2n)135\dots(2n-1).$$

For small n the encoding of exceptional permutations of type (1) is

$$\begin{aligned}
n = 1 \Rightarrow &\pi = 21 \Rightarrow E(\pi) = 21\\
n = 2 \Rightarrow &\pi = 2413 \Rightarrow E(\pi) = 2311\\
n = 3 \Rightarrow &\pi = 246135 \Rightarrow E(\pi) = 234111\\
n = 4 \Rightarrow &\pi = 24681357 \Rightarrow E(\pi) = 23451111
\end{aligned}$$

and for any n the encoding is

$$\pi = 24\dots(2n)13\dots(2n-1) \Rightarrow E(\pi) = 234\dots(n+1)111\dots1.$$

The exceptional permutations of type (2) have the form

$$(2n-1)(2n-3)\dots31(2n)(2n-2)\dots42.$$

For small n the encoding of exceptional permutations of type (2) is

$$\begin{aligned}
n = 1 \Rightarrow & \pi = 12 \Rightarrow E(\pi) = 11 \\
n = 2 \Rightarrow & \pi = 3142 \Rightarrow E(\pi) = 3121 \\
n = 3 \Rightarrow & \pi = 531642 \Rightarrow E(\pi) = 531321 \\
n = 4 \Rightarrow & \pi = 75318642 \Rightarrow E(\pi) = 75314321
\end{aligned}$$

and for any n the encoding is

$$\pi = (2n-1)(2n-3)\ldots 1(2n)(2n-2)\ldots 2 \Rightarrow E(\pi) = (2n-1)(2n-3)(2n-5)\ldots 1n(n-1)(n-2)\ldots 1.$$

The exceptional permutations of type (3) have the form

$$(n+1)1(n+2)2(n+3)3\ldots(2n)n.$$

For small n the encoding of exceptional permutations of type (3) is

$$\begin{aligned}
n = 1 \Rightarrow & \pi = 21 \Rightarrow E(\pi) = 21 \\
n = 2 \Rightarrow & \pi = 3142 \Rightarrow E(\pi) = 3121 \\
n = 3 \Rightarrow & \pi = 415263 \Rightarrow E(\pi) = 413121 \\
n = 4 \Rightarrow & \pi = 51627384 \Rightarrow E(\pi) = 51413121
\end{aligned}$$

and for any n the encoding is

$$\pi = (n+1)1(n+2)2\ldots(2n)n \Rightarrow E(\pi) = (n+1)1n1(n-1)1\ldots 21.$$

Lastly the exceptional permutations of type (4) have the form

$$n(2n)(n-1)(2n-1)(n-2)(2n-2)\ldots 1(n+1).$$

For small n the encoding of exceptional permutations of type (4) is

$$\begin{aligned}
n = 1 \Rightarrow & \pi = 12 \Rightarrow E(\pi) = 11 \\
n = 2 \Rightarrow & \pi = 2413 \Rightarrow E(\pi) = 2311 \\
n = 3 \Rightarrow & \pi = 362514 \Rightarrow E(\pi) = 352311 \\
n = 4 \Rightarrow & \pi = 48372615 \Rightarrow E(\pi) = 47352311
\end{aligned}$$

and for any n the encoding is

$$\pi = n(2n)(n-1)(2n-1)\ldots 1(n+1) \Rightarrow E(\pi) = n(2n-1)(n-1)(2n-3)(n-2)\ldots 11.$$

If we limit these languages to be over the alphabet $\{1, \ldots, k\}$, then $n \leq k$. Thus the number of exceptional permutations in Ω_k is finite. $\square$

Corollary 38. *The language of exceptional permutations of Ω_k under the rank encoding is regular.*

In conclusion, we have found that the set of rank encoded simple permutations with rank at most k is a regular language. Similarly the set of non-simple permutations with rank at most k is regular under the rank encoding. Additionally, the set of exceptional permutations, which are also simple, is in fact finite if we limit the permutations to a maximal rank k and so this finite set of rank encoded permutations is also a regular language.

Overall, we can now find the set of simple permutations within a regular class. But that still does not answer the question whether we can find all simple permutations of any class. Being able to do so would, amongst other results, simplify working with wreath closed classes as shown in corollary 45. We will see in the next section also why making a distinction between simple non-exceptional and simple exceptional permutations is necessary and useful.

3.8 Language of Simple Permutations of Non-Regular Pattern Classes

We are now going to look at classes $\mathcal{C}$ that do not have a regular language under the rank encoding. Our goal for this section is to find the set of simple permutations of $\mathcal{C}$. In this section we will denote $E_k(\mathcal{C})$ to be the language of all permutations of the class $\mathcal{C}$ that are rank encoded with highest rank k. So the corresponding set of permutations $\mathcal{C}_k$ to $E_k(\mathcal{C})$ is a subset of $\mathcal{C}$, if $\mathcal{C} \nsubseteq \Omega_k$. Further we will denote the set of all simple permutations that have rank at most k as $\mathcal{S}_k$. As before, $Si(\mathcal{C})$ is the set of all simple permutations of the class $\mathcal{C}$. Note that $Si(E_k(\mathcal{C})) = E_k(\mathcal{C}) \cap E_k(\mathcal{S}_k)$.

In [PR12] Pierrot and Rossin discuss the chains of simple permutations that are created through one or two point deletions of elements. These chains are based on the work of Schmerl and Trotter [ST93] and their results are narrowed down to simple permutations.

In a simple non-exceptional permutation π there is a subpermutation σ that is also simple, where $|\pi| - 1 = |\sigma|$. Further, if π is exceptional there is an exceptional permutation σ of the same type such that $\sigma \preceq \pi$ and $|\pi| - 2 = |\sigma|$.

Using this notion, which can be extended to chains of simple permutations, we are going to find the finite set of simple permutations in a pattern class that is not regular under the rank encoding.

Remark 39. The two point deletion on any type of exceptional permutation π means that the maximal letter of the rank encoded word corresponding to the contained exceptional permutation σ is smaller than the maximal letter of the rank encoded word corresponding to the original permutation.

Example. *Let us have a look at the exceptional permutations and their encodings of length* 8 *and the resulting permutations and encodings of the two point deletion.*

$$\pi = 24681357 \Rightarrow \pi' = 246135$$
$$E(\pi) = 23451111 \Rightarrow E(\pi') = 234111$$

$$\pi = 75318642 \Rightarrow \pi' = 531642$$
$$E(\pi) = 75314321 \Rightarrow E(\pi') = 531321$$

$$\pi = 51627385 \Rightarrow \pi' = 415263$$
$$E(\pi) = 51413121 \Rightarrow E(\pi') = 413121$$

$$\pi = 48372615 \Rightarrow \pi' = 362514$$
$$E(\pi) = 48352311 \Rightarrow E(\pi') = 352311$$

The following plots show the effect of the two-point deletion on each type of exceptional permutation. We can observe that the rank of the resulting exceptional permutation is lower, as we are removing the maximal points.

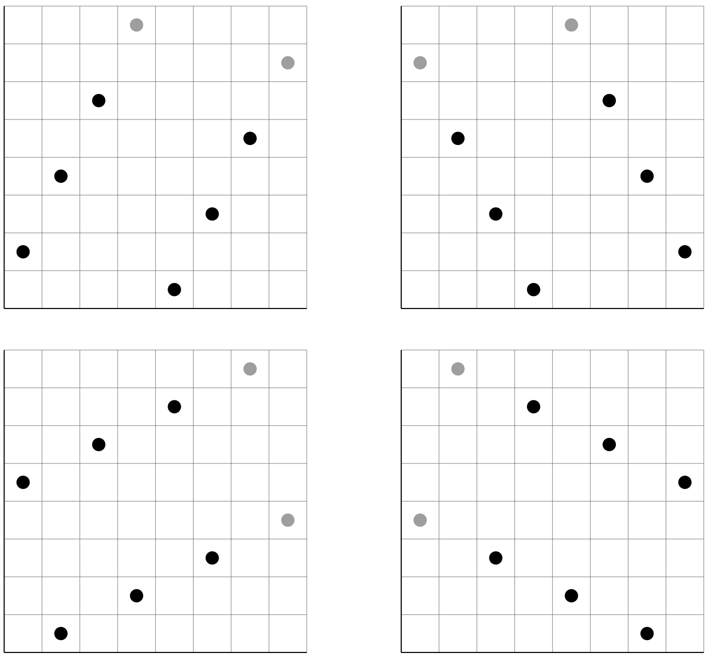


Figure 3.18: Plots of the four different types of exceptional permutations with the grey points indicating two-point deletion.

Theorem 40. *Let $\mathcal{C}$ be a permutation class. Let $Si(\mathcal{C})$ be the set of all simple permutations of $\mathcal{C}$. If $Si(E_k(\mathcal{C})) = Si(E_{k+1}(\mathcal{C})) = Si(E_{k+2}(\mathcal{C}))$ then $Si(E_k(\mathcal{C}))$ is the set of words corresponding to $Si(\mathcal{C})$.*

Proof. The assumption $Si(E_k(\mathcal{C})) = Si(E_{k+1}(\mathcal{C})) = Si(E_{k+2}(\mathcal{C}))$ says that there are no simple permutations of rank $k+1$ or $k+2$ in $\mathcal{C}$.

Take $\pi \in Si(\mathcal{C})$, $\pi \notin Si(E_k(\mathcal{C}))$ to have minimal rank $l > k$. So by the assumption $l \geq k+3$. There are two cases to consider.

If π is exceptional, then the exceptional permutation σ that results from the two point deletion on π will have rank $k < \text{rank}(\sigma) < l$. But l, the rank of $\pi \in Si(\mathcal{C})$, was minimal $> k$.

If π is not exceptional, then we can build a chain of simple permutations through point deletion. This chain will contain either an exceptional permutation π' with $\text{rank}(\pi') = l$ or a simple permutation σ with $\text{rank}(\sigma) = l-1$. For π' see the argument above on exceptional permutations. With σ not exceptional we are contradicting the assumption that $l > k$ is the minimal rank, where we have simple permutations that are not in $Si(E_k(\mathcal{C}))$ but in the class $\mathcal{C}$.

Thus, if $Si(E_k(\mathcal{C})) = Si(E_{k+1}(\mathcal{C})) = Si(E_{k+2}(\mathcal{C}))$ then we have found the whole set of simple permutations of the class $\mathcal{C}$ and this set is regular by corollary 36. $\square$

Remark 41. We can weaken the assumption of the above theorem to $Si(E_k(\mathcal{C})) = Si(E_{k+1}(\mathcal{C})) \supseteq Ex(E_{k+2}(\mathcal{C}))$, where $Ex(E_n(\mathcal{C}))$ $n > k$, is the set of rank encoded exceptional permutations of $\mathcal{C}$.

We have shown that for any pattern class we can find the set of all simple permutations, by finding a sufficiently large rank k, which is the maximal rank of all simple permutations of this class, and results in the regular language of all simple permutations of the class.

3.9 Regularity of Wreath Closed Classes

Definitions. *wreath product*

The *wreath product* of two classes $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$\mathcal{A} \wr \mathcal{B} = \mathcal{A}[\mathcal{B}] = \{\alpha[\beta_1, \ldots, \beta_n] : \alpha \in \mathcal{A},\ \beta_i \in \mathcal{B}\}.$$

Lemma 42. *Let $\mathcal{A}$ and $\mathcal{B}$ be regular classes under the rank encoding. Then $\mathcal{A} \wr \mathcal{B}$ is regular under the rank encoding if and only if $\mathcal{B}$ is finite or $\mathcal{A} \subseteq Av(21)$.*

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be regular classes and assume $\mathcal{A} \wr \mathcal{B}$ is also regular under the rank encoding.

If $\mathcal{B}$ is infinite and $21 \in \mathcal{A}$, then let $\mathcal{C} = 21[\mathcal{B}, \mathcal{B}] \subseteq \mathcal{A} \wr \mathcal{B}$. This language will have the following form

$$E(\mathcal{C}) = E(\mathcal{B})|_1^\infty\, E(\mathcal{B})$$

as $\mathcal{B}$ is infinite, the shift of the first $E(\mathcal{B})$ is unbounded, thus we have unbounded rank and so $E(\mathcal{C})$ is not regular. Recall that $\mathcal{L}|_i^j$ is the union of shifts of a language $\mathcal{L}$ by shifts between i and j inclusive.

If $\mathcal{B}$ is infinite and $\mathcal{A} = Av(21)$ then

$$E(\mathcal{A} \wr \mathcal{B}) = E(B)^*,$$

which is a regular language, as $E(\mathcal{A}) = 1^*$ and so the wreath product is a concatenation of permutations of $\mathcal{B}$.

If $\mathcal{B}$ is infinite and $\mathcal{A} \subset Av(21)$ then as $\mathcal{A}$ is finite and consists of strictly increasing permutations up to length say n, the language of the wreath product is a concatenation of n copies of $E(\mathcal{B})$.

Now let us assume that $\mathcal{A} \subseteq \Omega_k$ is a regular class and $\mathcal{B}$ is finite. Further, let $l = \max(|\beta|, \beta \in \mathcal{B})$ and $\pi = \alpha[\beta_1, \ldots, \beta_n] \in \mathcal{A} \wr \mathcal{B}$.

We will construct a transducer T such that

$$E(\mathcal{A})T = (E(\mathcal{A})T_1)T_2 = E(\mathcal{A} \wr \mathcal{B})$$

by constructing two smaller transducers.

The states of the first transducer consist of the pairs of gap sizes of α and π. The transitions of this transducer keep track of the ranks read in α and according to each rank and its position outputs all possible placements of blocks of different sizes in π. The output alphabet consists of a pair of letters (η, x) where η will then in the second transducer be translated to be the shift of the language of all possible blocks of length x.

So, the transducer T_1 has

input alphabet $\{1, \dots, k\}$

output alphabet $\{(\eta, x) : \eta \in \{1, \dots, k(l-1)\}, x \in \{1, \dots, l\}\}$

states $\{(gs'_\alpha(w), gs'_\pi(v)) : gs'_\alpha(w)$ gap sizes of prefix of $\alpha \in \mathcal{A}$, $gs'_\pi(v)$ gap sizes of $\pi \in \mathcal{A} \wr \mathcal{B}\}$

start state $s_1 = (\langle\emptyset\rangle, \langle\emptyset\rangle)$

accept states $A = \{(\langle\emptyset\rangle, \langle\emptyset\rangle), (\langle 0\rangle, \langle 0\rangle)\}$

The non-deterministic transition function for a given state and input is computed by algorithm 2. This algorithm follows the idea of constructing the transitions of the gap automaton $\mathcal{G}(k, A)$ as shown in algorithm 1.

Algorithm 2 Calculate the set of transitions of T_1 from state $(gs'_\alpha(w), gs'_\pi(v))$ with rank r.

```
Input: A state in form of gap sizes $(gs'_\alpha(w), gs'_\pi(v))$ $gs'_\alpha(w) = \langle g_1, \dots, g_x\rangle$, $gs'_\pi(v) = \langle e_1, \dots, e_x\rangle$
    and a letter $r$.
1:  $\Delta \leftarrow \emptyset$
2:  if $r \leq \sum gs'_\alpha(w)$ then                    ▷ The new block/rank goes into a gap
3:      Find the least $i \in \{1, \dots, x\}$ such that $r \leq g_1 + \cdots + g_i$
4:      for $t \in \{1, \dots, min(l, e_i)\}$ do                    ▷ Possible sizes of blocks
5:          if $r = g_1 + \cdots + g_i$ then                    ▷ New block at the top of the gap
                ▷ $(\eta, t)$ will be the output symbol and $(h, f)$ the gap sizes of the destination state
6:              $h \leftarrow gs'_\alpha(w)$
7:              $h_i \leftarrow g_i - 1$
8:              if ($h_i = 0$ and $t \neq e_i$) or $t > e_i - h_i$ then
9:                  Fail
10:             else
11:                 $f \leftarrow gs'_\pi(v)$
12:                 $f_i \leftarrow e_i - t$
13:                 $\eta \leftarrow f_1 + \cdots + f_{i-1}$
14:                 if $0 \in h$ and $|h| > 1$ then
15:                     Remove all 0's from $h$ and from $f$
16:                 end if
17:                 $\Delta \leftarrow \Delta \cup \{(\eta, t), (h, f)\}$
18:             end if
19:         else                    ▷ $r < g_1 + \cdots + g_i$
20:             $h \leftarrow gs'_\alpha(w)$
21:             Insert new element $r - (g_1 + \cdots + g_{i-1}) - 1$ into $h$ at position $i$
22:             $h_{i+1} \leftarrow g_i - h_{i-1} - 1$
23:             if $h_i = 0$ and $t \leq e_i - h_{i+1}$ then
24:                 $f \leftarrow gs'_\pi(v)$
25:                 Insert 0 to $f$ at position $i$
26:                 $f_{i+1} \leftarrow e_i - t$
27:                 $\eta \leftarrow f_1 + \cdots + f_{i-1}$
```

```
                if $0 \in h$ **and** $|h| > 1$ **then**
                    Remove all 0's from $h$ and from $f$
                **end if**
                $\Delta \leftarrow \Delta \cup \{(\eta, t), (h, f)\}$
            **else if** $h_i \neq 0$ **and** $t \leq e_i - (h_i + h_{i+1})$ **then**
                **for** $j \in \{h_i, \dots, e_i - t - h_i\}$ **do**        ▷ Possible sizes of the gap $h_i$ in $\pi$
                    $f \leftarrow gs'_\pi(v)$
                    Insert $j$ to $f$ at position $i$
                    $f_{i+1} \leftarrow e_i - t - j$
                    $\eta \leftarrow f_1 + f_2 + \dots + f_i$
                    $\Delta \leftarrow \Delta \cup \{(\eta, t), (h, f)\}$
                **end for**
            **else**
                Fail
            **end if**
        **end if**
    **end for**
**else**                                                    ▷ New maximal rank/block
    $h \leftarrow gs'_\alpha(w)$
    $h_{x+1} \leftarrow r - \sum gs'_\alpha(w) - 1$
    **if** $h_{x+1} = 0$ **then**
        $f \leftarrow gs'_\pi(v)$
        $f_{x+1} \leftarrow 0$
        $\eta \leftarrow \sum f$
        **if** $0 \in h$ **and** $|h| > 1$ **then**
            Remove all 0's from $h$ and from $f$
        **end if**
        **for** $t \in \{1, \dots, l\}$ **do**
            $\Delta \leftarrow \Delta \cup \{(\eta, t), (h, f)\}$
        **end for**
    **else**
        **for** $j \in \{h_{x+1}, \dots, l\}$ **do**
            $f \leftarrow gs'_\pi(v)$
            $f_{x+1} \leftarrow j$
            $\eta \leftarrow \sum f$
            **for** $t \in \{1, \dots, l\}$ **do**
                $\Delta \leftarrow \Delta \cup \{(\eta, t), (h, f)\}$
            **end for**
        **end for**
    **end if**
**end if**
**return** $\Delta$ the set of transitions
```

The second transducer T_2 has input alphabet $\{(\eta, x) : \eta \in \{1, \dots, k(l-1)\}, x \in \{1, \dots, l\}\}$ and output alphabet $\{1, \dots, kl\}$. For each letter (η, x) we non-deterministically output the whole language of the set of rank encoded and shifted words $\eta + E(\mathcal{B})^{(x)}$ of permutations of length x of $\mathcal{B}$. Clearly, as $E(\mathcal{A}) \subseteq \Omega_k$ is a regular language and the transducers T_1 and T_2 are finite state transducers, the output language is also regular.

Now we want to show that the output language is indeed $E(\mathcal{A} \wr \mathcal{B})$. Let $\pi = \alpha[\beta_1, \ldots, \beta_n] \in \mathcal{A} \wr \mathcal{B}$, with $E(\pi) = w = w_1 \ldots w_n$, where the $w_i = \eta_i + E(\beta_i)$ are factors and $|\beta_i| = x_i \in \mathbb{N}$. Then $(\alpha)T_1$ is included in the sequence (η_i, x_i) and $((\alpha)T_1)T_2$ includes a concatenation of $\eta_i + E(\mathcal{B})^{(x_i)}$ which clearly contains π.

Now we want to show that the language $(E(\mathcal{A})T_1)T_2$ excludes rank encoded permutations that are not in $\mathcal{A} \wr \mathcal{B}$. Let $\pi \notin \mathcal{A} \wr \mathcal{B}$ be such a permutation with $E(\pi) = w \in (E(\mathcal{A})T_1)T_2$. Then w consists of factors $w_1 \ldots w_n$, where one of the possible sequences of factors lies in $(\eta_i, x_i) \in (w)T_2^t$, so the w_i correspond to $\eta_i + E(\beta_i)$ where $|\beta_i| = x_i$, $\beta_i \in \mathcal{B}$, and the sequence $(\eta_i, x_i) \in (E(\mathcal{A}))T_1$. It is now easy to check that in fact $w = E(\alpha[\beta_1, \ldots, \beta_n])$ for $\alpha \in \mathcal{A}$ and $\beta_i \in \mathcal{B}$. So $\pi = \alpha[\beta_1, \ldots, \beta_n]$ which contradicts our assumption.

Thus we have found that the language of the wreath product of the regular class $\mathcal{A}$ and the finite class $\mathcal{B}$ is

$$E(\mathcal{A} \wr \mathcal{B}) = (E(\mathcal{A})T_1)T_2 \subseteq E(\Omega_{kl})$$

and regular. □

Definitions. *wreath closed class, wreath closure*

A class $\mathcal{A}$ is said to be *wreath closed* if $\mathcal{A} = \mathcal{A} \wr \mathcal{A}$ and the *wreath closure* of a class $\mathcal{A}$ is

$$\langle \mathcal{A} \rangle = \bigcup_{n=1}^{\infty} \mathcal{A}_n,$$

where $\mathcal{A} = \mathcal{A}_1$, $\mathcal{A}_n = \mathcal{A} \wr \mathcal{A}_{n-1}$. The wreath closure is the smallest wreath closed set of permutations that contains $\mathcal{A}$.

Lemma 43. *The wreath closure of a regular class $\mathcal{A}$ containing the permutation* 21 *is not regular.*

Proof. Let $\mathcal{A} = \{21\}$, then the wreath closure of $\mathcal{A}$ is the class of descending permutations. As $\mathcal{A}$ is finite, it is a regular language under the rank encoding. But the class of all descending permutations is not regular under the rank encoding as the language is over an infinite alphabet. □

Corollary 44. *A wreath closed class $\mathcal{A}$ is regular under the rank encoding if and only if $\mathcal{A}$ is finite or consists of ascending permutations.*

Let us summarise our findings, the wreath product of two regular classes is a regular language under the rank encoding if and only if the second class is finite or 21 is not a permutation of the first class. Next we have shown that the wreath closure of a class is not a regular language under the rank encoding if the class contains 21. Finally, through all theses proofs we came to the conclusion that a class is wreath closed and regular if and only if the class is finite or the class avoids 21.

So we have not been able to use regular languages to characterise wreath closed classes. But below is a corollary from [AA05] which uses the wreath closure of the set of simple permutations of a class to represent the wreath closed class.

Corollary 45 ([AA05]). *Let $\mathcal{A}$ be a wreath closed class. Then*

$$\mathcal{A} = \langle Si(\mathcal{A}) \rangle,$$

where $Si(\mathcal{A})$ is the set of simple permutations of $\mathcal{A}$.

As we know that we can find the set of simple permutations of a class, we can utilise that mechanism to check whether a given class is wreath closed.

Chapter 4

Implementations

We have created implementations of previously published, but unanalysed and new algorithms within the `GAP` [GAP15] language. The collection of these algorithms can be found in the `PatternClass` package [ALH12]. This package works on `GAP` 4.6.4 and higher.

All algorithms in the following sections are experimentally investigated for their time complexity. We will first concentrate on algorithms of some of the work presented in the previous sections, before showing testing of implementations of a couple important functions of the `PatternClass` package [ALH12].

The `PatternClass` can be found under the URL in [ALH12] and is also attached as an electronic appendix to this thesis.

4.1 New Algorithms

Permutations will be stored in these implementations as arrays with unique entries. Similar treatment will be given to words. Regular languages will be represented as automata, which recognise the language. Automata will be represented similarly to before as the quintuple $(\Sigma, S, \delta, s_1, A)$, where Σ is the alphabet (set of letters), S the set of states, δ the transition function $S \times \Sigma \to S$, $s_1 \in S$ the start state and $A \subseteq S$ the set of final states. The transition function δ will be represented as a matrix of sets where the rows are labelled with the letters of the alphabet, the columns are labelled with states and the entries $\delta_{as} = \delta(s, a) \subseteq S$ for $a \in \Sigma, s \in S$.

Remark 46. For practical reasons we will represent the transition matrix of deterministic automata as sets of singletons.

Example. *The following automaton is an example of a non-deterministic automaton.*

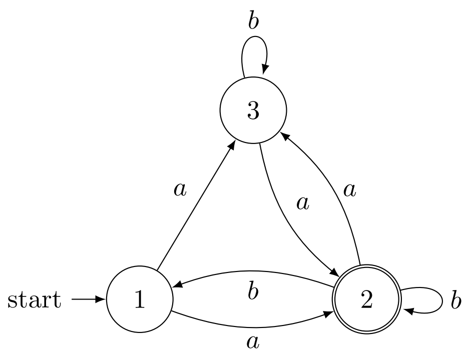


Figure 4.1: Example of a non-deterministic automaton

The alphabet is $\Sigma = \{a, b\}$, the set of states $S = \{1, 2, 3\}$, the start state is 1, the set of accept states is $\{2\}$ and the transition function

$$\delta = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \{2,3\} & \{3\} & \{2\} \\ \emptyset & \{1,2\} & \{3\} \end{pmatrix} \end{array}.$$

Another example, this time of a deterministic automaton, is the gap automaton for $k = 3$, as shown in figure 3.11. There the alphabet is $\Sigma = \{1, 2, 3\}$, the set of states $S = \{\langle\emptyset\rangle, \langle 0\rangle, \langle 1\rangle, \langle 2\rangle, \langle 1, 1\rangle\}$, the start state is $\langle\emptyset\rangle$, the set of accept states $A = \emptyset$ and the transition function is

$$\delta = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccc} \langle\emptyset\rangle & \langle 0\rangle & \langle 1\rangle & \langle 2\rangle & \langle 1,1\rangle \end{array} \\ \begin{pmatrix} \{\langle 0\rangle\} & \{\langle 0\rangle\} & \{\langle 0\rangle\} & \{\langle 1\rangle\} & \{\langle 1\rangle\} \\ \{\langle 1\rangle\} & \{\langle 1\rangle\} & \{\langle 1\rangle\} & \{\langle 1\rangle\} & \{\langle 1\rangle\} \\ \{\langle 2\rangle\} & \{\langle 2\rangle\} & \{\langle 1,1\rangle\} & \{\langle 2\rangle\} & \{\langle 1,1\rangle\} \end{pmatrix} \end{array}.$$

Any algorithms working on constructing automata will be, through the way the automata are defined, spending the least time writing the transition matrix. Thus most of our complexities will be corresponding to the sizes of the transition matrices of the output automata. Further we are utilising the several library functions provided from the `GAP` packages `Automata` [DLM11] and `PatternClass`, which are standard automaton operations.

UnionAutomata Returns the automaton recognising the union of the languages which are accepted by the input automata. The complexity of the function is linear with respect to the size of the transition function of the output automaton. The algorithm is based on the basic proof of the closure of the union of regular languages [Sip96].

IntersectionAutomata Returns the automaton recognising the language of the intersection of the languages accepted by the input automata. The complexity is linear to the size of the transition function of the output automaton. This is because the output automaton is generated by running through both input automata in parallel, for more details see [HMU06].

ReversedAutomaton The returned automaton accepts the reversed language of the input automaton. The complexity of the function is linear with respect to the transition function of the output automaton, which is generated by algorithm 3, with complexity $\mathcal{O}(n^2k)$ for non-deterministic automata and $\mathcal{O}(nk)$ for deterministic automata, where $n = |S|$ and $k = |\Sigma|$.

A proof of the regularity of the reversed language, based on regular expressions can be found in [HMU06].

Algorithm 3 Generate the reversed transition function δ' of a given automaton.

```
Input: Automaton $(\Sigma, S, \delta, s_1, A)$
1: $\delta' \leftarrow |\Sigma| \times |S|$ matrix of $\emptyset$
2: for $a \in \Sigma$ do
3:     for $s \in S$ do
4:         for $s' \in \delta_{a,s}$ do
5:             Add $\{s\}$ to $\delta'_{a,s'}$
6:         end for
7:     end for
8: end for
   return $\delta'$
```

ProductOfLanguages The output automaton accepts the language consisting of the concatenation of the languages of the input automata. As the automaton is generated by adding epsilon transitions from the accept states of the first input automaton to the start state of the following automaton, complexity of the algorithm is linear with respect to the size of the transition function of the output automaton. A more detailed description of the algorithm can be found in the proof of the closure of regular languages under concatenation in [Sip96].

All CPU times were obtained on a machine with thirty two 2.3 GHz AMD Opteron$^{\text{TM}}$Processor 6376 with hyperthreading while running on just one of them and 512GB RAM. The experiments were run with `GAP` 4.6.4 and version 1.12358 of `PatternClass` [ALH12].

4.1.1 Block-Decomposition

In this section we will again only consider plus- and minus-indecomposable (*pmi*) permutations. The algorithm presented will find the unique block-decomposition $\sigma[\alpha_1, \dots, \alpha_m] = \pi$, corresponding to any pmi permutation. This algorithm conforms with the theory in section 3.5.

There are two more library functions used in the algorithms below. First, *IsInterval*(s), which takes a sequence s and checks whether its values are contiguous. This function is based on the idea of finding common intervals between two permutations as described in [UY00]. As pointed out in [BCdMR08] the more complicated algorithms presented in [UY00], when applied to random permutations of moderate size, can be slower than the basic $\mathcal{O}(n^2)$ algorithm. *IsInterval* uses the basic algorithm from [UY00], with complexity $\mathcal{O}(n^2)$, to compare the sequence s to the identity permutation of the same length. As we are not trying to find all common intervals, *IsInterval* utilises the comparison part of the algorithm once on the whole sequence s. This modification results in *IsInterval* having complexity $\mathcal{O}(n)$.

Secondly the function *Sortex*(s), which takes in an sequence s of comparable elements and finds the permutation π such that s^π is sorted. The sequence s may be a sequence of numbers or of sequences, which we compare lexicographically. It is worth noting that any sorting algorithm can easily be extended to an algorithm for *Sortex* with the same complexity.

In algorithm 4 we calculate all maximal intervals of a permutation π. Every time we enter the outer loop we know that at the end of line 5 we have already found maximal intervals containing the point 1 up to $i-1$. We now search backwards for the end point of the maximal interval containing i. Overall the runtime of algorithm 4 is $\mathcal{O}(n^3)$.

Algorithm 4 Find all maximal intervals in π.

```
Input: Permutation $\pi$
$i \leftarrow 1$
$a \leftarrow [\,]$
$l \leftarrow |\pi|$
while $i \leq l$ do
    $j \leftarrow l$
    while $j \geq i$ do
        if IsInterval($\pi(i) \dots \pi(j)$) and ($i \neq 1$ or $j \neq l$) then
            Add $\pi(i) \dots \pi(j)$ to $a$
            break
        else
            $j \leftarrow j - 1$
        end if
    end while
    $i \leftarrow j + 1$
end while
return $a$
```

Algorithm 5 takes the unrefined sequence of intervals that is output in algorithm 4 and by using the *Sortex* function, calculates the permutation that corresponds to the placement of the disjoint intervals in p.

Algorithm 5 Find the permutation corresponding to the sequence of sequences.

```
Input: Sequence $a$ of maximal intervals
simp $\leftarrow$Sortex($a$)
return simp
```

Lastly algorithm 6 takes the unrefined intervals individually from the output sequence of algorithm 4 and finds the order isomorphic permutation corresponding to that interval, which is the permutation that sorts the interval.

Algorithm 6 Turn a duplicate-free sequence into its order isomorphic permutation.

```
Input: Duplicate free sequence $\alpha$
$\alpha \leftarrow$Sortex($\alpha$)
return $\alpha$
```

The overall computation of the unique block-decomposition $\sigma[\alpha_1, \dots, \alpha_m]$ of π requires one execution of algorithm 4, one of algorithm 5 and at most n executions of algorithm 6.

In summary the algorithm we have found to compute the block-decomposition of permutations has complexity $\mathcal{O}(n^3)$. This is because algorithm 4 has complexity $\mathcal{O}(n^3)$ and both algorithm 5 and 6 have complexity $\mathcal{O}(n \log(n))$, where $n = |\pi|$.

Our testing of the block-decomposition algorithm in `PatternClass` with the lengths of random permutations ranging between 100 and 7289 is consistent with the complexity of the algorithm being $\mathcal{O}(n^3)$.

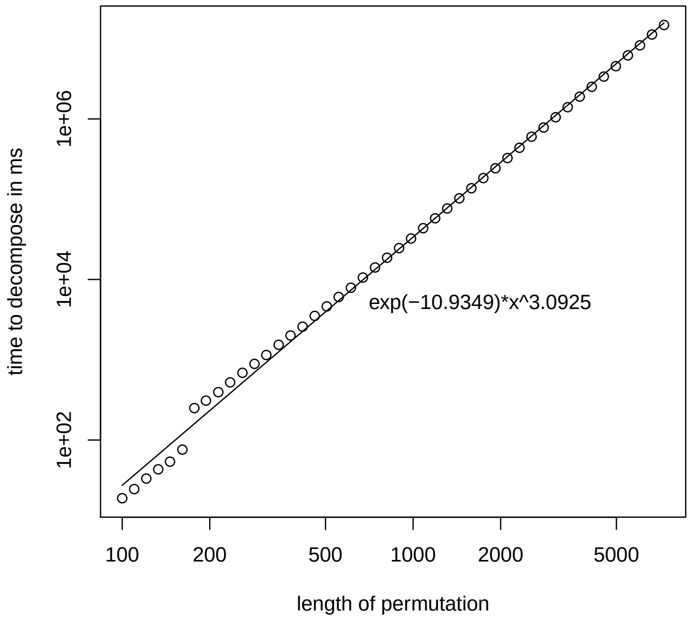


Figure 4.2: Plot with logarithmic axes of the test results for calculating the block-decomposition of a permutation.

The function $y = \exp(-10.9349)x^{3.0925}$ represents the best log-linear fit to the results, which is consistent with the time complexity of $\mathcal{O}(n^3)$.

4.1.2 Plus-Indecomposable Language

We will now describe the algorithm to build the automaton accepting the words corresponding to the plus-indecomposable permutations under the rank encoding. The approach we take is similar to the proof in section 3.2.

As a quick summary, the proof and algorithm to find the language of all plus-indecomposable permutations of $E_R(\Omega_k)$ is based on the idea that a plus-indecomposable permutation π cannot be composed of a concatenation of two words representing valid rank encoded permutations. We can construct the automaton that rejects such concatenated words.

Remark 47. The algorithm below uses the automaton accepting the language $E_R(\Omega_k)$ as the testing was done for this language. We can use *IntersectionAutomaton* to find $\mathcal{I}_P(\mathcal{C})$ for any regular $\mathcal{C}$ or in fact we can adapt the construction below to output this automaton directly.

Algorithm 7 Build the automaton accepting the rank encodings of plus-indecomposable permutations up to rank k.

```
Input: $k \in \mathbb{N}$
 1: Construct automaton $(\Sigma, S, \delta, s_1, A)$ accepting $E_R(\Omega_k)$
 2: $S' \leftarrow S \sqcup \{x, y\}$
 3: $A' \leftarrow A \sqcup \{x\}$
 4: Initialise $\delta'$ as $|\Sigma| \times |S'|$ matrix of $\emptyset$
 5: for $a \in \Sigma$ do
 6:     for $s \in S$ do
 7:         $\delta'_{a,s} \leftarrow \delta_{a,s}$
 8:     end for
 9:     $\delta'_{a,x} \leftarrow \delta_{a,s_1}$
10:     for $s \in A$ do
11:         $\delta'_{a,s} \leftarrow \{y\}$
12:     end for
13: end for
    return Automaton $(\Sigma, S', \delta', x, A')$
```

This algorithm copies the transition matrix of the original automaton and adds the two new states as well as changes a few transitions during the copying process. Overall the complexity is linear in the size of the transition function of the output automaton. As the transition function is a matrix the complexity is $\mathcal{O}(k^2)$, as the deterministic automaton of $E_R(\Omega_k)$, has an alphabet of size k and k states.

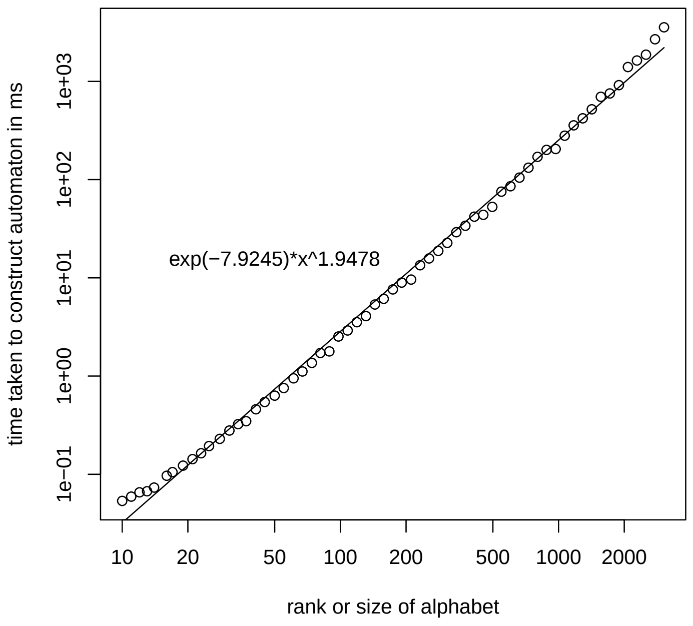


Figure 4.3: Plot with logarithmic axes of the test results for constructing the automaton accepting the language of rank encoded plus-indecomposable permutations up to rank k.

The function $y = \exp(-7.9245)x^{1.9478}$ represents the best log-linear fit to the result, which makes the implemented function in `PatternClass` consistent with the time complexity of $\mathcal{O}(k^2)$.

4.1.3 Minus-Decomposable Language

Following the proof in section 3.3, the language of rank encoded minus-decomposable permutations up to rank k is $E_R(\mathcal{D}_M) = \mathscr{L} \cap E_R(\Omega_k)$, where

$$\mathscr{L} = \bigcup_{d=1}^{k-1} L_d \text{ and } L_d = \left\{\{d+1,\dots,k\}^+\{1,\dots,d\}^d\right\}.$$

In algorithm 8 the procedure *LdkAut* represents the language L_d in reverse, where the procedure *LAut* represents the union of L_d over $d < k$, thus constructs the automaton accepting the reverse of the language $\mathscr{L}$. As we are working on permutations with rank encodings up to rank k, the alphabet of all automata below is of size k.

Algorithm 8 Build the automaton accepting the language $E_R(\mathcal{D}_M)$ of minus-decomposable rank encoded permutations.

```
1:  procedure LdkAut($d, k \in \mathbb{N}$)
2:      $\Sigma \leftarrow \{1,\dots,k\}$
3:      $S \leftarrow \{1,\dots,d+3\}$
4:      Initialise $\delta$ as a $|\Sigma| \times |S|$ matrix of $\emptyset$
5:      for $a \leftarrow 1, d$ do
6:          for $s \leftarrow S$ do
7:              if $s \leq d$ then
8:                  $\delta_{a,s} \leftarrow \{s+1\}$
9:              else
10:                 $\delta_{a,s} \leftarrow \{d+3\}$
11:             end if
12:         end for
13:     end for
14:     for $a \leftarrow d+1, k$ do
15:         for $s \leftarrow S$ do
16:             if $s \leq d$ or $s = d+3$ then
17:                 $\delta_{a,s} \leftarrow \{d+3\}$
18:             else
19:                 $\delta_{a,s} \leftarrow \{d+2\}$
20:             end if
21:         end for
22:     end for
        return Automaton $(\Sigma, S, \delta, 1, \{d+2\})$
23: end procedure
```

```
24: procedure LAut($k \in \mathbb{N}$)
25:     $\mathcal{A} \leftarrow$ LdkAut($1, k$)
26:     for $i \leftarrow 2, k-1$ do
27:         $\mathcal{B} \leftarrow$ LdkAut($i, k$)
28:         $\mathcal{A} \leftarrow$ UnionAutomata($\mathcal{A}, \mathcal{B}$)
29:     end for
    return Automaton $\mathcal{A}$
30: end procedure
31: $\mathcal{A} \leftarrow$ LAut($k$)
32: $\mathcal{A} \leftarrow$ ReversedAutomaton($\mathcal{A}$)
33: $\mathcal{B} \leftarrow$ Automaton accepting $E_R(\Omega_k)$
34: $\mathcal{A} \leftarrow$ IntersectionAutomaton($\mathcal{A}, \mathcal{B}$)
    return Automaton $\mathcal{A}$
```

The automaton constructed by *LdkAut* has $d+3$ states and $\delta_{a,s}$ for any letter a and any state s is a set of size 1. Thus the number of transitions is $k(d+3)$. So overall the complexity of *LdkAut* is $\mathcal{O}(k(d+3))$. The output automaton of *LAut* has an alphabet of size k and k^2 states, due to the repeated union of the automata of *LdkAut*. As all entries of the transition matrix of *LdkAut* are singleton sets, the transitions of the automaton constructed by *LAut* are also sets of size 1. So the automaton has k^3 transitions. Thus the complexity of *LAut* is $\mathcal{O}(k^3)$, this complexity is not exceeded by the $k-1$ calls of *UnionAutomata*.

Finally the output automaton of *LAut* is reversed and then intersected with the automaton accepting the language $E_R(\Omega_k)$, which has an alphabet of size k, k states and k^2 transitions, as it is a deterministic automaton. The intersection of these two automata is a non-deterministic automaton with k letters, k^3 states and k^5 transitions. So the overall complexity of algorithm 8 lies in the construction of this intersection, which has complexity $\mathcal{O}(k^5)$.

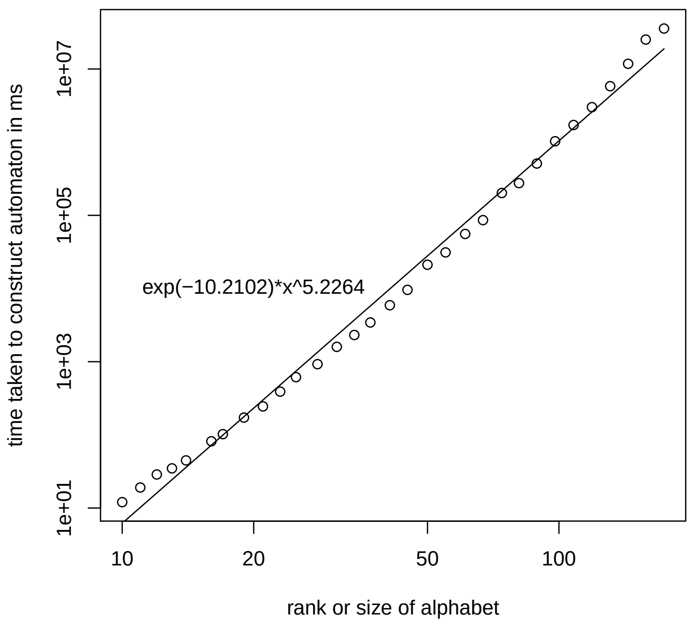


Figure 4.4: Plot with logarithmic axes of the test results of the construction of the automata accepting the rank encodings of minus-decomposable permutations up to rank k.

The function $y = \exp(-10.2102)x^{5.2264}$ represents the best log-linear fit to the results, which is consistent with the time complexity of $\mathcal{O}(k^5)$.

4.1.4 Non-Simple Language

As described in section 3.7 the language of all simple permutations of Ω_k under the rank encoding is:

$$\begin{aligned} E_R(\mathcal{NS}_k) =& E_R(\Omega_k) \cap \Bigg(\bigcup_{l=1}^{k-1} \mathscr{P}_l \bigcup_{m=l}^{k-1} m + E_R(\hat{\Omega}_{k-m}) \cup \bigcup_{j=1}^{k-1} j + E_R(\hat{\Omega}_{k-j}) \cup \\ & \cup \bigcup_{a=2}^{k-1} \bigcup_{b=0}^{k-1-a} \mathscr{Q}_{a,b} \bigcup_{i=0}^{a-2} ((b+i) + E_R(\hat{\Omega}_{k-(b+i)}))^{(a-i)} \cup E_R(\Omega_k \setminus \{\varepsilon\}) E_R(\Omega_k \setminus \{\varepsilon\}) \Bigg) \Sigma^*, \end{aligned} \tag{4.1}$$

where

Σ is the alphabet $\{1, \ldots, k\}$, $k \in \mathbb{N}$, $k \geq 3$.

$\mathscr{P}_l$ is the language of prefixes of rank encoded permutations, where $\sum gs'(w) = l$.

$\mathscr{Q}_{i,j}$ is the language of prefixes of rank encoded permutations, where in $gs'(w)$ there is a gap g_x of size i and sum of the gaps $g_1, \ldots g_{x-1}$ equals to j.

$i + E_R(\Omega_{k-i})$ is the language of $E_R(\Omega_{k-i})$, $i \in \mathbb{N}$, with the alphabet shifted upwards by i.

$E_R(\Omega_k)^{(i)}$ is the sublanguage of $E_R(\Omega_k)$ containing the words of length $\leq i$, $i \in \mathbb{N}$.

$E_R(\hat{\Omega}_k)$ is the sublanguage of $E_R(\Omega_k)$ containing the words of length > 1.

Towards the end of algorithm 9 we can find the automaton of the above language being constructed. The loop starting at line 75 corresponds to $\bigcup_{l=1}^{k-1} \mathscr{P}_l \bigcup_{m=l}^{k-1} m + E_R(\hat{\Omega}_{k-m})$, the loop starting at line 80 corresponds to $\bigcup_{j=1}^{k-1} j + E_R(\hat{\Omega}_{k-j})$ and the nested loops starting at line 85 correspond to $\bigcup_{a=2}^{k-1} \bigcup_{b=0}^{k-1-a} \mathscr{Q}_{a,b} \bigcup_{i=0}^{a-2} ((b+i) + E_R(\hat{\Omega}_{k-(b+i)}))^{(a-i)}$. Finally line 97, the automaton accepting rank encoded plus-decomposable permutations corresponds to $E_R(\Omega_k \setminus \{\varepsilon\}) E_R(\Omega_k \setminus \{\varepsilon\})$.

Algorithm 9 Build the automaton accepting the language of non-simple rank encoded permutations up to rank k.

```
 1: procedure NextGap($gs'(w) = \langle g_1, \dots, g_n \rangle, r \in \mathbb{N}$)
                                   ▷ This procedure is algorithm 1 in section 3.7.
 2: end procedure

 3: procedure GapAut($k$)
        ▷ GapAut returns the gap automaton $\mathcal{G}(k, \emptyset)$ for any $k \in \mathbb{N}$ as described in section 3.7.
 4:     $\Sigma \leftarrow \{1, \dots, k\}$
 5:     $S \leftarrow \{\langle \emptyset \rangle\}$
 6:     $\delta \leftarrow (\emptyset)$
 7:     $i \leftarrow 1$
 8:     while $i \le |S|$ do
 9:         $tmp \leftarrow (\emptyset)$
10:         for $r \leftarrow 1, k$ do
11:             $s \leftarrow$ NextGap($s_i, r$)
12:             if $s \notin S$ then
13:                 Add $s$ to $S$
14:             end if
15:             Add $\{s\}$ to $tmp$
16:         end for
17:         Add $tmp$ as a column to $\delta$
18:         $i \leftarrow i + 1$
19:     end while
    return Automaton $(\Sigma, S, \delta, s_1, \emptyset)$
20: end procedure

21: procedure SumAut($sum \in \mathbb{N}, \mathcal{A}$)
                      ▷ $\mathcal{A} = (\Sigma, S, \delta, s_1, \emptyset)$, is the gap automaton $\mathcal{G}$ for $k \in \mathbb{N}$
22:     $A' \leftarrow \{\emptyset\}$
23:     for $s \in S$ do
24:         if $\sum s = sum$ then
25:             Add $s$ to $A'$
26:         end if
27:     end for
    return Automaton $(\Sigma, S, \delta, s_1, A')$
28: end procedure
```

```
procedure GapSumAut($g_x, sum \in \mathbb{N}, \mathcal{A}$)
                    ▷ $\mathcal{A} = (\Sigma, S, \delta, s_1, \emptyset)$, is the gap automaton $\mathcal{G}$ for $k \in \mathbb{N}$
    $A' \leftarrow \{\emptyset\}$
    for $s = \langle g_1, \dots, g_n \rangle \in S$ do
        if $g_x \in s$ then
            $tmp \leftarrow$ {all positions of occurrences of $g_x$ in $s$}
            for $i \in tmp$ do
                if $i = 1$ and $sum = 0$ then
                    Add $s$ to $A'$
                else if $sum = \sum_{a=1}^{i-1} g_a$ then
                    Add $s$ to $A'$
                end if
            end for
        end if
    end for
    return Automaton $(\Sigma, S, \delta, s_1, A')$
end procedure

procedure LengthBoundAut($min, max \in \mathbb{N}, \mathcal{A}$)
                    ▷ $\mathcal{A}$ is an automaton with $(\Sigma, S, \delta, s_1, A)$
    $S' \leftarrow \{1, \dots, max + 2\}$
    Initialise $\delta'$ as $|\Sigma| \times |S'|$ matrix of $\emptyset$
    for $a \in \Sigma$ do
        Let the $a$-th row of $\delta'$ be of the form $(\{2\}, \{3\}, \dots, \{max+1\}, \{max+2\}, \{max+2\})$
    end for
    $\mathcal{B} \leftarrow (\Sigma, S', \delta', 1, \{min + 1, \dots, max + 1\})$
    return IntersectionAutomaton($\mathcal{A}, \mathcal{B}$)
end procedure

procedure ZeroOneAutomaton($k \in \mathbb{N}$)
    $\Sigma \leftarrow \{1, \dots, k\}$
    $S \leftarrow \{0, 1, 2\}$
    $s_1 \leftarrow 0$
    $A \leftarrow \{2\}$
    $\delta \leftarrow \begin{array}{cc} & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \Sigma^* & \begin{pmatrix} \{1\} & \{2\} & \{2\} \end{pmatrix} \end{array}$
    return Automaton $(\Sigma, S, \delta, s_1, A)$
end procedure
```

```
59: procedure ShiftAut($x, k \in \mathbb{N}$)
60:     $\mathcal{A} \leftarrow$ automaton accepting $E_R(\Omega_{k-x})$
61:     $\mathcal{B} \leftarrow$ ZeroOneAutomaton($k - x$)
62:     $\mathcal{C} \leftarrow$ IntersectionAutomaton($\mathcal{A}, \mathcal{B}$)        $\triangleright\ \mathcal{C} = (\Sigma, S, \delta, s_1, A)$
63:     Initialise $\delta'$ as $k \times |S|$ matrix of $\emptyset$
64:     for $a \in 1, k$ do
65:         for $s \in S$ do
66:             if $a > x$ then
67:                 $\delta'_{as} \leftarrow \delta_{a-x,s}$
68:             end if
69:         end for
70:     end for
    return Automaton ($\{1, \ldots, k\}, S, \delta, s_1, A$)
71: end procedure

72: Initialise $\mathcal{A}, \mathcal{B}$ to be automata accepting the empty language
73: $\mathcal{G} \leftarrow$ GapAut($k$)
74: for $i \leftarrow k - 1, 1$ do
75:     $\mathcal{A} \leftarrow$ UnionAutomata($\mathcal{A}$,ShiftAut($i, k$))
76:     $\mathcal{C} \leftarrow$ ProductOfLanguages(SumAut($i, k, \mathcal{G}$),$\mathcal{A}$)
77:     $\mathcal{B} \leftarrow$ UnionAutomata($\mathcal{B}, \mathcal{C}$)
78: end for
79: for $i \leftarrow 1, k - 1$ do
80:     $\mathcal{A} \leftarrow$ ShiftAut($i, k$)
81:     $\mathcal{B} \leftarrow$ UnionAutomata($\mathcal{B}, \mathcal{A}$)
82: end for
83: Initialise $\mathcal{C}$ to be the automaton accepting the empty language
84: for $a \leftarrow 2, k - 1$ do
85:     for $b \leftarrow 0, k - a - 1$ do
86:         $\mathcal{A} \leftarrow$ GapSumAut($a, b, k, \mathcal{G}$)
87:         for $i \leftarrow 0, a - 2$ do
88:             $\mathcal{D} \leftarrow$ ShiftAut($b + i, k$)
89:             $\mathcal{D} \leftarrow$ LengthBoundAut($2, a - i, k, \mathcal{D}$)
90:             $\mathcal{E} \leftarrow$ ProductOfLanguages($\mathcal{A}, \mathcal{D}$)
91:             $\mathcal{C} \leftarrow$ UnionAutomata($\mathcal{C}, \mathcal{E}$)
92:         end for
93:     end for
94: end for
95: $\mathcal{C} \leftarrow$ ProductOfLanguages($\mathcal{C}, \{1, \ldots, k\}^*$)
96: $\mathcal{A} \leftarrow$ Automaton of PlusDecomposableAut($k$)
97: $\mathcal{D} \leftarrow$ UnionAutomata($\mathcal{C}, \mathcal{A}$)
98: $\mathcal{E} \leftarrow$ UnionAutomata($\mathcal{D}, \mathcal{B}$)
99: $\mathcal{B} \leftarrow$ automaton accepting $E_R(\Omega_k)$
    return The automaton returned by IntersectionAutomaton($\mathcal{E}, \mathcal{B}$).
```

The procedure *NextGap* is algorithm 1 as shown in section 3.7. It takes gap sizes $gs'(w) = \langle g_1, \dots, g_n \rangle$ and a letter $r \in \mathbb{N}$ and calculates the next gap sizes for the word wr. The complexity of this procedure is linear to the length of the gap sizes $gs'(w)$. So the complexity is $\mathcal{O}(k)$.

The procedure *GapAut* builds the gap automaton construct. Each state of this automaton is described by gap sizes and the transitions are determined by *NextGap*. The set of gap sizes for k is equivalent to the set of compositions of $k-1$, thus the set of states is of size $2^{k-1}+1$. As we are working over an alphabet of size k, the overall size of the transition matrix of the automaton output by *GapAut* is $k(2^{k-1}+1)$, thus the complexity of *GapAut* is $\mathcal{O}(k2^k)$.

Next the procedure *SumAut* takes the automaton output by *GapAut*, or any automaton with the states represented as gap sizes and determines the set of accept states, by finding the gap sizes $gs'(w) = \langle g_1, \dots, g_n \rangle$ such that $\sum gs'(w) = \sum_{i=1}^{n} g_i = sum$. For that each state has to be checked. As the input automaton is the gap automaton construct from section 3.7 and returned by *GapAut* the number of states is $2^{k-1}+1$, thus the complexity of *SumAut* is $\mathcal{O}(2^k)$, whereas the output automaton is the same as the input automaton except for the set of accept states. The language accepted by *SumAut* is $\mathscr{P}_l$, for $l = sum$, from equation (4.1).

Similarly for *GapSumAut* we are looking at the input automaton, which has its states described as gap sizes. Each state is checked to see whether it contains a gap size g_x and whether the preceding gap sizes sum to sum, if so this state is added to the set of accept states. Thus the time complexity of *GapSumAut* is $\mathcal{O}(2^k)$, when the input automaton is the gap automaton and the output automaton is the same except for the set of accept states. The language accepted by this output automaton is $\mathscr{Q}_{i,j}$ from equation (4.1).

LengthBoundAut constructs an automaton, which accepts words of lengths between and inclusive min and max, from the input automaton. The implementation constructs an automaton that accepts all words over the alphabet of the input automaton between the lengths min and max and then intersects the resulting automaton with the input. The intersection is the most time costly function of this procedure. The input automaton has an alphabet of size $|\Sigma| = k$, $|S|$ states and $k|S|$ transitions. The constructed automaton has k letters, $max+2$ states and $k(max+2)$ transition. So the output automaton consists of k letters, $|S|(max+2)$ states and $k^2|S|(max+2)$ transitions. Thus the complexity of *LengthBoundAut* is $\mathcal{O}(k^2\, max)$.

The procedure *ZeroOneAutomaton* is an auxiliary procedure that creates an automaton over the alphabet $\Sigma = \{1, \dots, k\}$ that accepts the language of all words over Σ of length > 1. The constructed automaton has 3 states and $3k$ transitions, hence the complexity of the procedure is $\mathcal{O}(k)$.

The automaton constructed by the procedure *ShiftAut* accepts the language $i + E_R(\hat{\Omega}_{k-x})$ from equation (4.1). The output automaton is based on the automaton accepting $E_R(\Omega_{k-x})$. First the words of length ≤ 1 are excluded before the transition matrix is shifted to k letters and the first $x-1$ rows are empty transitions. The automaton resulting from the intersection of $E_R(\Omega_{k-x})$ with the automaton output by *ZeroOneAutomaton* has $k-x$ letters, $3(k-x)$ states and $3(k-x)^3$ transitions. As we are passing through all the transitions to shift them to a $k \times 3(k-x)$ matrix, each transition has to be accessed. So the complexity of *ShiftAut* is $\mathcal{O}(k^4)$.

Constructing the automaton in the loop starting at line 75 will have complexity $\mathcal{O}(k^2 2^k)$ and the loop starting at line 80 increases the complexity to $\mathcal{O}(k^3 2^k)$. Further the nested loops starting at line 85 increase the complexity of the algorithm to $\mathcal{O}(k^4 2^k)$.

Finally the union of the automata resulting from the above loops with the automaton accepting rank encoded plus-decomposable permutations and the intersection with the automaton accepting the language $E_R(\Omega_k)$ results in an algorithm with time complexity $\mathcal{O}(k^5 2^k)$.

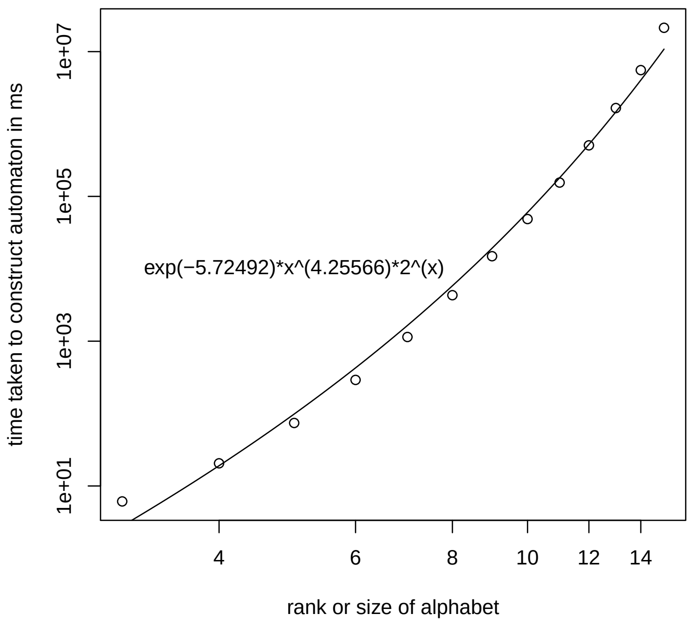


Figure 4.5: Plot of test results for constructing the language of non-simple permutation over rank k which is the same as the size of the alphabet of the automaton against time.

The function $y = \exp(-5.72492)x^{4.2556}2^x$ represents the best fit to the results, which is faster than the calculated time complexity of $\mathcal{O}(k^5 2^k)$. This is due to the fact that we are using some heuristics to replace the large non-deterministic automaton by a smaller equivalent automaton, which still accepts the same language. These reductions are not indicated in the pseudo-algorithms, but are happening in the implementation. In particular the reductions are done on lines 78, 82, 92 and 99, after the union of the two automata on those lines.

4.2 Analysis of Known Algorithms

The `PatternClass` package started of as an aid to work with known classes under the rank encoding and their languages. The original approach taken is through token passing networks and their natural properties to create permutation pattern classes which are regular under the rank encoding, as described in section 2.2. One can create the language of the class by inputting the TPN that describes said class, or more directly the language of the class or the basis. In [AAR03] it is extensively discussed on how to calculate the language of the basis directly from the language of the class or vice versa.

As previously shown the construction of the language of the basis $E_R(B)$ from the class $\mathcal{C}$ is based on the equation

$$E_R(B) = (E_R(\mathcal{C}))^C \cap ((E_R(\mathcal{C}))^C \mathcal{D}^t)^C, \tag{4.2}$$

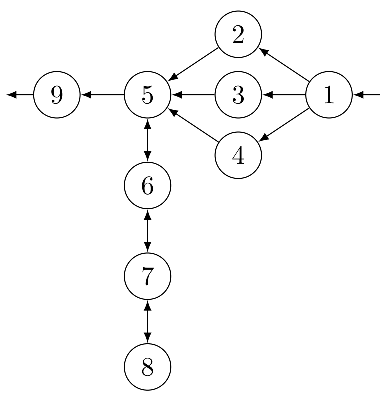
where $\mathcal{D}$ is a transducer that deletes an arbitrary letter in a rank encoded permutation, and returns a word that represents the permutation that had the point removed that corresponds to the removed letter [AAR03].

It has been shown in [AAR03] that it is also possible to move from knowing the language of the basis to the language of the class under the rank encoding. By using an involvement transducer $\mathcal{H}$ that removes any number of letters from the input word.

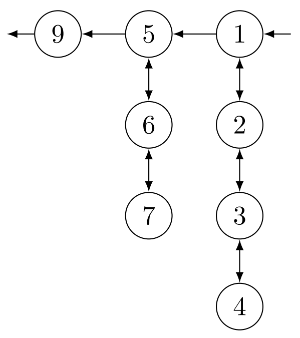
$$E_R(\mathcal{C}) = (E_R(B)\mathcal{H}^t)^C \cap E_R(\Omega_k). \tag{4.3}$$

We test the implementation of both equations (4.2) and (4.3) against 4 different types of languages over a range of ranks. Three of the languages are based on types of TPNs and the last language is $E_R(\Omega_k)$.

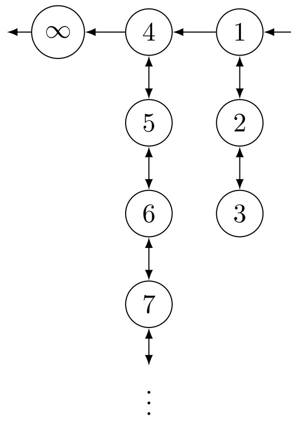
The first set of tests are done over the languages constructed by a TPN that contains a finite buffer and a finite stack. An example TPN is shown in figure 4.6a. The second type of TPN consists of two finite stacks in sequence, see figure 4.6b, and the third type is two stacks in sequence, the first has finite size, whereas the second is infinite, but we limit the number of tokens at any time in the network, which gives us the finite alphabet and a maximal rank, see figure 4.6c for an example.



(a) TPN with a buffer of size 3 and a stack of size 4.



(b) TPN with two finite stacks in sequence.



(c) TPN with a finite stack followed by an infinite stack.

For the testing of the functions *BasisAutomaton* and *ClassAutomaton* on the languages of the classes we do the following. First we apply and time the function *BasisAutomaton* which computes the language of the basis of the class as shown in equation (4.2). Then we apply *ClassAutomaton* to the resulting languages and measure the time it takes to compute the languages of the classes from the bases using equation (4.3). Finally for an additional test, which resulted from curiosity, we used the same method as above on the language of Ω_k. So first applied *BasisAutomaton* and then to the result *ClassAutomaton* and timed each function. But we also applied *ClassAutomaton* directly to the language of Ω_k, which results in the final class being the empty class.

For each type of TPN we used different sizes of data structures within the graphs and averaged the time taken for each rank of the resulting languages of the class.

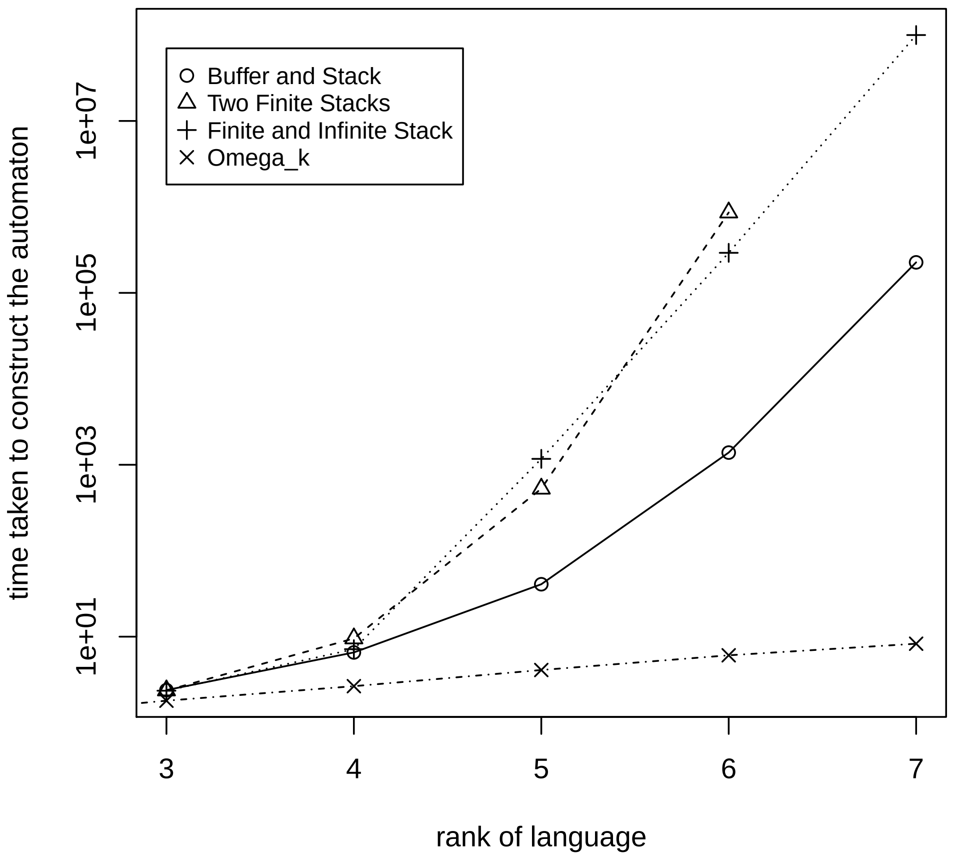


Figure 4.7: Plot of run time of *BasisAutomaton* when applied to languages of different types of TPNs and Ω_k.

We can see that the function *BasisAutomaton* is taking exponential time corresponding to the rank of the input language, also with each change to the structures of the TPN the time increases as the underlying languages and automata get larger and more complicated.

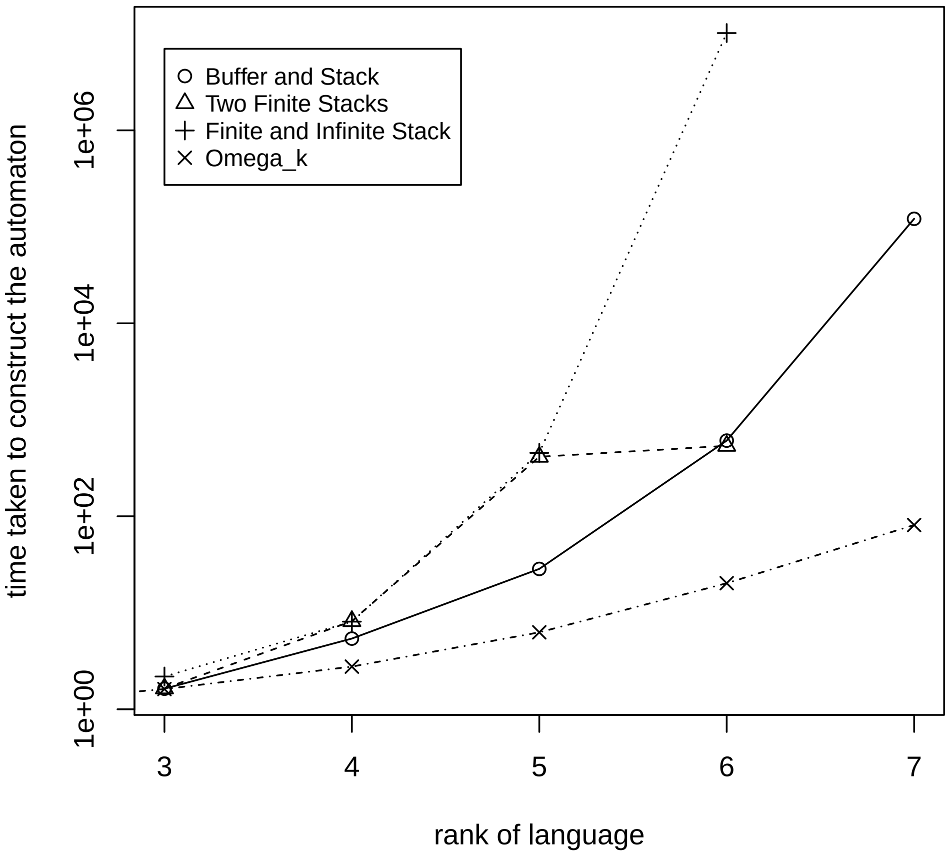


Figure 4.8: Plot of run time of *ClassAutomaton* when applied to languages of the bases of different types of TPNs and the basis of Ω_k.

We have similar results for *ClassAutomaton*. It seems that at rank 6 the time taken to construct the language of the class created by the TPNs consisting of two finite stacks is faster. This is due to the fact that for more complicated TPNs of this type with rank 6 the construction of the final automaton involves steps in which automata are determinised, and this has the effect that these automata have an extremely large number of states. This lead to the memory space being used up and we could not calculate the final automaton, but seeing the trends within the other constructions, we can predict with some certainty that the average time for TPNs of that form with rank 6 is longer.

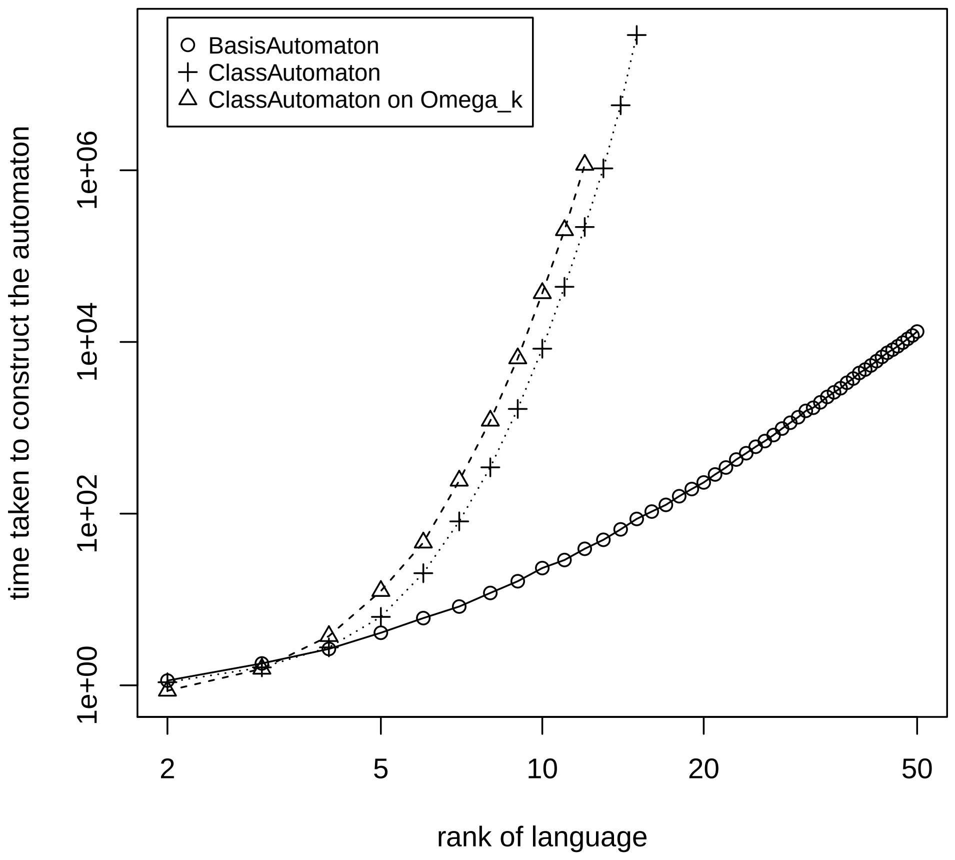


Figure 4.9: Plot of run time of *BasisAutomaton* and *ClassAutomaton* when applied to $E_R(\Omega_k)$.

It is interesting to see that finding the language of the class $Av(\Omega_k)$ takes little longer than finding the language of Ω_k knowing the basis is empty. This is most likely due to the fact that the involvement transducer will create a non-deterministic automaton with a high number of states and which when determinised will be even larger.

A similar observation can be made when comparing the runtimes of *ClassAutomaton* to *BasisAutomaton*. This has to do with the fact that the deletion transducer of rank k has fewer states than the involvement transducer of rank k. Thus the application of the former transducer to languages will result in a smaller automaton than the latter.

4.3 Conclusions

We have shown that the main proofs of regular languages of sets of permutations of regular classes can be implemented and that the code conforms with the complexities of the algorithms. In particular we have shown that the time complexity to find the block-decomposition of a permutation of length n is $\mathcal{O}(n^3)$. The construction of the automaton accepting rank encoded words of the plus-indecomposable permutations of a regular class with maximal rank k has complexity $\mathcal{O}(k^2)$. The time complexity to construct the language of the rank encoded minus-decomposable permutations of a regular class with maximal rank k is $\mathcal{O}(k^5)$. Finally, the construction of the regular language of simple permutations under the rank encoding with maximal rank k has complexity $\mathcal{O}(k^5 2^k)$.

Furthermore we have shown some testing of the essential functions that calculate the rank encoded language of the basis of a regular class knowing the language of the class and vice versa.

The above functions are the core functions of the `PatternClass` package. There are more functions, amongst which there are functions which construct the non-deterministic automaton representing the rank encoded regular languages of permutations generated by any token passing network; functions for the rank encoding of a permutation and the rank decoding of a word; functions for checking whether a permutations is simple, plus- or minus-decomposable or whether a sequence is an interval; a function that constructs the language of the direct sum of two regular classes and for any k a function that calculates the language of exceptional permutations. Additionally there are also functions that calculate the set of simple permutations resulting from the point deletion on simple permutations, these functions are useful when looking at the chains of simple permutations.

Chapter 5

Grid Class Encoding

This chapter is dedicated to the work on finding the basis of a geometric grid class through the language of the encoding of geometric grid classes using the cell alphabet which is introduced in section 2.6.

We would like to make the reader aware that this chapter is an attempt at finding a constructive way to find the language of the basis, while starting with the language of the geometric grid class. In [AAB$^+$11b] it is shown that every geometric grid class is finitely based and so the language of the basis under the geometric grid class encoding is regular. To show that the partial well order property of the classes is utilised as well as Higman's Theorem. This method is not constructive as Higman's Theorem does not have a constructive proof yet.

Our aim is to find a constructive and implementable way to be able to go back and forth between a class and its basis, through their languages, similarly as it is done with the rank encoding in [AAR03]. Further the two attempts below partially follow the proof in [AAB$^+$11b].

Definitions. *finite state transducer, offset matrix, offset cells/entries, one point extension, normal language, standard language*

Here we introduce two attempts at alternative proofs that are more constructive and which should allow for an implementation. For both, the notion of transducers is relevant. As a quick reminder a *finite state transducer* is a type of finite automaton with output strings. Thus it is a sextuple $(\Sigma, \Gamma, S, \delta, s_1, A)$, where Σ is the input alphabet, Γ is the output alphabet, S is the finite set of states, δ is the transition function $S \times (\Sigma \times \Gamma) \to S$, $s_1 \in S$ is the start state and $A \subseteq S$ is the set of accept states.

Let $\mathcal{C}$ be a geometric grid class over a partial multiplication matrix M. We can extend M to its *offset matrix* M^{+1} by extending the entries M_{ij} of M by 5×5 submatrices $(M_{ij})^{+1}$ of M^{+1}, as follows. If $M_{ij} = 0$, then the submatrix $(M_{ij})^{+1}$ is,

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

if $M_{ij} = 1$ then we extend it to the following $(M_{ij})^{+1}$

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 1 & 0 & 0 & 0 & 0 \end{matrix}$$

and if $M_{ij} = -1$, then $(M_{ij})^{+1}$ is

$$\begin{matrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1. \end{matrix}$$

We call the cells/entries in M^{+1} that are above indicated by bold $\mathbf{1}$'s *offset cells/entries*. Let $\pi \in \mathcal{C}$ with $|\pi| = n$, then $\hat{\pi} \notin \mathcal{C}$, $|\hat{\pi}| = n+1$ is an *one point extension* of π if $\hat{\pi}$ contains an *offset point* p that lies in an offset cell and if p is removed from $\hat{\pi}$ the resulting permutation $\pi \in \mathcal{C}$. We will denote $\mathcal{C}^{+1}$ to be the set of permutations that contains the permutations of $\mathcal{C}$ with an offset point.

Further we will be using the multi-valued encoding, $E_G(\pi)$, introduced in [AAB$^+$11b]. So $E_G(\pi)$ is the encoding over the alphabet $\Sigma = \{a_{ij} : M_{ij} \neq 0\}$, with respect to the row and column signs of the partial multiplication matrix M. An encoding of a permutation is *normal* if it is lexicographically least with respect to the trace monoid (see section 7 of [AAB$^+$11b]). So every gridding with respect to M of that permutation has a unique normal encoding. We denote this encoding by $\overline{E}_G$, which is still multi-valued per permutation. An encoding of a permutation is *standard* if it is lexicographically least amongst all normal encoding of the same permutation, so we are choosing the word with the "lexicographically least" gridding of each permutation. We will denote this encoding by $\dot{E}_G$. In [AAB$^+$11b] Albert et al. have shown that both languages $\overline{E}_G(\mathcal{C})$ and $\dot{E}_G(\mathcal{C})$ are regular languages.

Our first approach gives us a language that contains the basis elements, but we have additional permutations that are not in the basis or the class in this language.

Proposition 48. *Let $\mathcal{C}$ be a geometric grid class of M. The set of geometric grid encoded basis elements of $\mathcal{C}$ with respect to the matrix M^{+1} is included in*

$$E_G^{M^{+1}}(\mathcal{X}) = \dot{\mathcal{E}}^{M^{+1}}(\mathcal{C}^{+1}) \setminus \dot{E}_G^{+1}(\mathcal{C}).$$

Before proving the above proposition, we have a working example to introduce notation.

Example. *Let $\mathcal{C} = Av(21)$, so $\mathcal{C}$ is griddable by the matrix $M = (1)$. M has row and column signs $r_1 = c_1 = 1$ and the cell alphabet with respect to M is $\Sigma = \{a_{11}\}$.*

The multi-valued language of the class $\mathcal{C}$ under the grid encoding with respect to M is $E_G(\mathcal{C}) = a_{11}^*$. We normalise the language to avoid having words containing the same letters in different order representing the same permutation, we will still have multiple words representing a permutation. The normal language of $E_G(\mathcal{C})$ is $\overline{E}_G(\mathcal{C}) = a_{11}^*$. Let us now extend this language to the language *over the offset matrix M^{+1} of M,*

$$M^{+1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

M^{+1} has row and column signs $r_i = c_i = 1$ for $i \in \{1, \ldots, 5\}$ and the cell alphabet is $\Sigma^{+1} = \{a_{11}, a_{24}, a_{33}, a_{42}, a_{55}\}$. We are now extending $\overline{E}_G(\mathcal{C})$ to a language over the alphabet and gridding of M^{+1} while still respecting the gridding and lexicographical order of M. The resulting language $\mathcal{E}^{+1}(\mathcal{C})$ contains the normal pre-images of $\overline{E}_G(\mathcal{C})$ with respect to M^{+1},

$$\mathcal{E}^{+1}(\mathcal{C}) = (a_{11}|a_{24}|a_{55})^*|(a_{11}|a_{33}|a_{55})^*|(a_{11}|a_{42}|a_{55})^*.$$

Next we will find the language of one-point extended permutations of $\mathcal{C}$ under the grid encoding. We will use the language $\mathcal{E}^{+1}(\mathcal{C})$. We know that the one point extensions of the permutations griddable by a matrix contain a point lying out with the line segment in the cells. Thus the language will be

$$\mathcal{E}^{+1}(\mathcal{C}^{+1}) = (a_{11}|a_{24}|a_{55})^*(a_{33}|a_{42})|(a_{11}|a_{33}|a_{55})^*(a_{24}|a_{42})|(a_{11}|a_{42}|a_{55})^*(a_{24}|a_{33}).$$

Let us standardise this language,

$$\dot{\mathcal{E}}^{M^{+1}}(\mathcal{C}^{+1}) = \dot{E}_G^{+1}(\mathcal{C}^{+1}) = a_{11}^*|(a_{11}^* a_{24} a_{33} a_{55}^*)$$

and the standard language of $\mathcal{C}$ with respect to M^{+1} is

$$\dot{E}_G^{M^{+1}}(\mathcal{C}) = a_{11}^*.$$

Now the language containing the basis amongst other permutations not in the class is

$$E_G^{M^{+1}}(\mathcal{X}) = \dot{\mathcal{E}}^{M^{+1}}(\mathcal{C}^{+1}) \setminus \dot{E}_G^{+1}(\mathcal{C}) = a_{11}^* a_{24} a_{33} a_{55}^*.$$

This is a regular language as it is represented by a regular expression, but further all the languages above are regular.

As we can see in the above example, we have found an infinite regular language that contains the basis element, in the example $\pi = 21$, $E_G^{M^{+1}}(\pi) = a_{24}a_{33}$. Currently we have not found a way to extract the basis elements from this language without having prior knowledge of the basis or the permutations that correspond to the words in the language.

Proof of proposition 48. Below is a general idea of the proof setup, as seen in the example.

$$\begin{tikzcd}
 & E_G(\mathcal{C}) \arrow[d, "(1)"', "\text{normalise}"] & \\
 & \overline{E}_G(\mathcal{C}) \arrow[d, "(2)"', "\text{extend}"] & \\
 & \text{All pre-images of } \overline{E}_G(\mathcal{C}) \text{ with respect to } M^{+1},\ \mathcal{E}^{+1}(\mathcal{C}) \arrow[dl, "\text{standardise}"', "(3)"] \arrow[dr, "(4)"', "\text{one point extension}"] & \\
\dot{\mathcal{E}}^{M^{+1}}(\mathcal{C}) = \dot{E}_G^{+1}(\mathcal{C}) \arrow[ddr] & & \mathcal{E}^{+1}(\mathcal{C}^{+1}) \arrow[d, "(5)"', "\text{standardise}"] \\
 & & \dot{\mathcal{E}}^{+1}(\mathcal{C}^{+1}) = \dot{E}_G^{M^{+1}}(\mathcal{C}^{+1}) \arrow[dl, "\text{(6) difference}"'] \\
 & E_G^{M^{+1}}(\mathcal{X}) &
\end{tikzcd}$$

Let $\mathcal{C}$ be a geometrically griddable class over the matrix M and the set of non-uniquely encoded permutations of $\mathcal{C}$ with respect to M be $E_G(\mathcal{C})$.

Step (1) normalises the language $E_G(\mathcal{C})$ to $\overline{E}_G(\mathcal{C})$ by using the idea of trace monoids as described in [AAB$^+$11b] in the proof of proposition 7.2. So each word in $\overline{E}_G(\mathcal{C})$ is the lexicographically least word of each gridding of the permutations in $\mathcal{C}$ and $\overline{E}_G(\mathcal{C})$ is regular.

Step (2) expands the language $\overline{E}_G(\mathcal{C})$ from over the cell alphabet with respect to M to all possible encodings over M^{+1} while still considering the grid lines of M. This is done with a finite state transducer that takes the alphabet of M and each letter a_{ij} is translated to the letters of the corresponding submatrix of M^{+1} while still considering the order of how the points are added, which is dictated by the row and column signs. This means that the encoded words in $\mathcal{E}^{+1}(\mathcal{C})$ are all ways of encoding the permutations in $\mathcal{C}$ over M^{+1} while still being lexicographically least with respect to the griddings over M and $\mathcal{E}^{+1}(\mathcal{C})$ is regular.

Step (3) standardises the words of $\mathcal{E}^{+1}(\mathcal{C})$ by using the idea of marking letters in the words that represent the same permutation when they witness the fact that they are indicating a shift in the grid lines. Out of those multiple words per permutation we choose the word representing the permutation with the gridding that has the grid lines as far to the right and as high as possible. For more explanation see section 8 of [AAB$^+$11b]. We now have a unique word with a gridding per permutation in the class $\mathcal{C}$ over the matrix M^{+1}, thus the resulting regular language $\dot{\mathcal{E}}^{+1}(\mathcal{C}) = \dot{E}_G^{M^{+1}}(\mathcal{C})$.

In Step (4) we add the offset points to the permutations that are represented by the words in $\mathcal{E}^{+1}(\mathcal{C})$. In the encoding we are adding a letter that represents an offset point. This procedure is done with a transducer, which for each word returns a set of words each word containing one offset point. The resulting language $\mathcal{E}^{+1}(\mathcal{C}^{+1})$ will include words that represent permutations in $\mathcal{C}$ as well as permutations not in $\mathcal{C}$, which are the one point extensions of the permutations in $\mathcal{C}$.

Step (5) standardises all words in $\mathcal{E}^{+1}(\mathcal{C}^{+1})$ using the same method as described in Step (3). Similarly as in Step (3) we now have a unique word per permutation with a chosen gridding $\dot{\mathcal{E}}^{+1}(\mathcal{C}^{+1}) = \dot{E}_G^{M^{+1}}(\mathcal{C}^{+1})$.

Finally we take the two languages resulting from Step (3) and (5) $\dot{E}_G^{M^{+1}}(\mathcal{C})$ and $\dot{E}_G^{M^{+1}}(\mathcal{C}^{+1})$

respectively, and take the difference. As $\dot{E}_G^{M^{+1}}(\mathcal{C}^{+1})$ contains the words representing one point extensions as well as permutations of $\mathcal{C}$, we are now excluding the words representing permutations of $\mathcal{C}$, the resulting set will contain the encoded words of the basis of $\mathcal{C}$. This language is regular as all the above languages are regular and regularity is preserved under set difference. □

Remark 49. The resulting language $E_G^{M^{+1}}(\mathcal{X})$ contains amongst other words the grid encoded permutations of the basis of $\mathcal{C}$. If a way of determining which words are indeed the words of the basis, without decoding, is found the above proof will be a constructive and implementable way to find the basis of a geometric grid class.

Remark 50. As there are only a finite number of offset points, there are only a finite number of choices for the one point extensions of the permutations in $\mathcal{C}$, thus the basis of $\mathcal{C}$ is finite, as stated in [AAB$^+$11b].

This might aid us in the search for the basis elements in the language $E_G^{M^{+1}}(\mathcal{X})$. As the language most likely will be always infinite.

Remark 51. We are unable to easily obtain a language over M^{+1} which represents all griddings of $\mathcal{C}$.

Now for a different way of attempting to get the language of the basis of a geometric grid class. We follow the idea of using the encodings of permutations not in the class $\mathcal{C}$. These permutations have an additional point, which prevents them from being in $\mathcal{C}$. The overall process to finding the basis using these types of permutations is similar to what was done with the rank encoding in [AAR03].

Conjecture 52. *Let M, N be two $0, \pm 1$ matrices, with* $\mathrm{Geom}(M) = \mathcal{C}$, $\mathrm{Geom}(N) = \mathcal{D}$. *Then $\overline{E}_G^{M \times N}(\mathcal{C} \cap \mathcal{D})$, the language of normal encodings of permutations in $\mathcal{C} \cap \mathcal{D}$, is regular.*

Assuming this conjecture holds we can then find the language of the basis under the geometric grid encoding.

Proposition 53. *Let $\mathcal{C}$ be a geometric grid class of M. The set of geometric grid encoded basis elements of $\mathcal{C}$ with respect to the matrix M^{+1} is*

$$E_G^{M^{+1}}(B) = \overline{(\mathcal{E}^C \mathcal{D}^T)}^C \setminus \overline{E}_G^{M^{+1}}(\mathcal{C}).$$

Proof. Again we start with a diagram showing the flow of the proof.

$$\begin{tikzcd}
E_G(\mathcal{C}) \arrow[d, "(1)"', "\text{extend}"] \\
E_G^{M\times M^{+1}}(\mathcal{C}\cap \mathrm{Geom}(M^{+1})) = E_G^{M\times M^{+1}}(\mathcal{C}) \arrow[d, "(2)"', "\text{choose language}"] \\
\mathcal{E} = \{v : (w,v) \in E_G^{M\times M^{+1}}(\mathcal{C})\} \arrow[d, "(3)"', "\text{complement}"] \\
\mathcal{E}^C \arrow[d, "(4)"', "\text{one point extend}"] \arrow[ddd, bend right=40, "\text{normalise set difference}"'] \\
\mathcal{E}^C\mathcal{D}^T = \mathcal{N} \arrow[d, "(5)"', "\text{complement}"] \\
\mathcal{N}^C \arrow[d, "\text{normalise set difference}"] \\
\overline{\mathcal{N}}^C \setminus \overline{\mathcal{E}}^C = E_G^{M^{+1}}(B) \arrow[d, "(7)"', "\text{standardise}"] \\
\dot{E}_G^{+1}(B)
\end{tikzcd}$$

Step (1) applies conjecture 52 to our class $\mathcal{C} \subseteq \mathrm{Geom}(M)$, and as M^{+1} is the extension matrix of M, $\mathcal{C} \subset \mathrm{Geom}(M)$. So $\mathrm{Geom}(M^{+1} \cap \mathcal{C}) = \mathcal{C}$. Thus, using conjecture 52

$$E_G^{M\times M^{+1}}(\mathcal{C}\cap \mathrm{Geom}(M^{+1})) = E_G^{M\times M^{+1}}(\mathcal{C})$$

is a regular multi-valued language. In step (2) we choose the words over M^{+1} that correspond to permutations in $\mathcal{C}$. The language $\mathcal{E}$ is still regular, as all the words that we have chosen are words of the form $(\varepsilon, w) \in E_G^{M\times M^{+1}}(\mathcal{C})$ which is a regular language, because this subset of words is accepted by the automaton of $E_G^{M\times M^{+1}}(\mathcal{C})$.

Step (3) takes the complement of $\mathcal{E}$. Regularity of languages is preserved under complementation.

In step (4) we add a letter to each word in our given language $\mathcal{E}^C$ by applying the transducer shown in figure 5.1. The alphabet that we are working on is $\Sigma^{M^{+1}}$ and the output language is still regular.

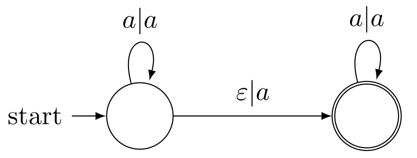


Figure 5.1: One point addition transducer, where $a \in \Sigma$ is any letter and Σ is any alphabet.

Step (5) takes the complement of the language again. At this point we have the multi-valued geometric grid encoding of permutations that are not in our original class $\mathcal{C}$.

Because we are still in the multi-valued encoding, let us normalise both languages $\mathcal{E}^C$ and $\mathcal{N}^C$. We have now the language of all permutations not in the class $\mathcal{C}$ but griddable by M^{+1} and all permutations that are not in $\mathcal{C}$ by one point and griddable by M^{+1}. To find the language of the basis we take the set difference in step (6). This will return the regular language of multi-valued encodings of permutations of the basis of $\mathcal{C}$ over M^{+1}. Thus in step (7) we standardise the language, to get the regular single-valued language of geometric grid encodings of permutations of the basis of $\mathcal{C}$. $\square$

We can see that the two proposed concepts use a similar approach to the language of the basis through the extended matrix M^{+1}. This idea stems from the proof used in [AAB$^+$11b]. The first approach attempts to avoid the use of Higman's Theorem at the end by using transducers to find the language containing the basis. What we need is a mechanism to eliminate all other permutations from the final language. This mechanism should be in place either within the current construct or after the language containing the basis has been found. The former thought brought on the idea of the second approach to finding the language of the basis. Unfortunately, this concept is dependent on the conjecture of finding a regular language in the intersection of languages of two classes over different matrices. So far, a counter example to conjecture 52 has not been found, using manual checking on small examples of pmm matrices. If a counter example exists, there is little chance that it will be contained in a high dimensional matrix as the small examples should cover all possibilities of encodings and positioning of points in the permutations and griddings.

Chapter 6

Conclusion

This thesis has shown a variety of language theoretic applications to permutations and permutation pattern classes, specifically, when looking at the rank encoding and the regular languages of it. We have seen that under this encoding the set of plus- and minus-(in)decomposable permutations in a regular class form a regular language and that the respective languages need a different approach even though the properties have the same origin and are similar. It was possible from there to find in a regular class the regular language of all encoded permutations in which the block-decomposition has the same simple permutation. This further opened the work into the language of simple permutations, for which a regular language was also found. With this language it is possible to find the whole set of simple permutations within a class that is not regular under the rank encoding. This result has high impact on the research of permutation pattern classes as simple permutations are regarded as building blocks for the understanding of pattern classes. For example, proposition 9 and corollary 10 use the set of simple permutations in a type of class that we have found not to be regular except for finite cases. Being able to find the set of simple permutations within a class easily, allows in these cases it to be simpler to find the properties of the class.

It was also shown that for a spectrum of different class types, such as separable classes, a regular language under the rank encoding cannot be found. Further research in the realm of the rank encoding is to find other sets of permutations with a property and to show whether these sets are regular under the rank encoding. Having an implementation of the rank encoding and the regular languages of sets of permutations as well as pattern classes under the rank encoding, gives access to compute specific examples, which could be of use to prove or disprove new research within the field. Giving access to a program that does these calculations and language theoretic constructions allows for extensive testing of current theories and further development of ideas.

Further other encodings of permutations were discussed. The insertion encoding, which can be implemented in two slightly different manners. One yields regular classes, which have similar properties to those regular classes under the rank encoding. The implementation of the regular encoding is more time consuming and its applications can get complex due to the way the encoding is defined. The focus in this thesis was not on the context-free insertion encoding as there is not much research within that field, nor many applications. It seems like the context-free aspect of the insertion encoding does not yield any more properties about specific sets of permutations, due to the encoding being non-unique to the permutations and similarly to the regular insertion encoding is time consuming in its implementation.

The geometric grid class encoding has left some open questions. In the process of attempting to find a constructive way to calculate the language of the encoded basis of a geometric grid class difficulties were found in the definition of the language which in the language theoretic context seems simple, but the underlying consequences for the permutations are convoluted. If a constructive method can be found for the process of finding the language of the basis, an implementation of this encoding would help the research of permutation pattern classes, due to the similarities of grid classes and wreath classes and the extensive cross-overs between grid classes and other combinatorial and algebraic fields.

Index of Definitions

Bibliography

[AA05] M H Albert and M D Atkinson, *Simple permutations and pattern restricted permutations*, Discrete Mathematics **300** (2005), no. 1–3, 1–15.

[AAAH01] Michael H Albert, Robert E L Aldred, Mike D Atkinson, and Derek A Holton, *Algorithms for Pattern Involvement in Permutations*, Algorithms and Computation, Lecture Notes in Computer Science, vol. 2223, Springer Berlin / Heidelberg, 2001, pp. 355–367.

[AAB11a] MH Albert, MD Atkinson, and R Brignall, *The enumeration of permutations avoiding 2143 and 4231*, Pure Mathematics and Applications (2011), 87–98.

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

[AAK03] M H Albert, M D Atkinson, and M Klazar, *The enumeration of simple permutations*, Journal of Integer Sequences **6** (2003), no. 4, 1–18.

[AAR03] M H Albert, M D Atkinson, and N Ruškuc, *Regular closed sets of permutations*, Theoretical Computer Science **306** (2003), no. 1–3, 85–100.

[AAV11] Michael H Albert, M D Atkinson, and Vincent Vatter, *Subclasses of the separable permutations*, Bulletin of the London Mathematical Society **43** (2011), no. 5, 859–870.

[ALH12] Michael Albert, Steve Linton, and Ruth Hoffmann, *PatternClass – Permutation Pattern Classes*, http://ruthh.host.cs.st-andrews.ac.uk/pkg.html, 2012.

[ALR04] M H Albert, S Linton, and N Ruškuc, *On the Permutational Power of Token Passing Networks*, Tech. report, Department of Computer Science, University of Otago, 2004.

[ALR05] Michael H Albert, Steve Linton, and Nik Ruškuc, *The insertion encoding of permutations*, Electron J Combin **12** (2005), no. 1, R47.

[ALT97] M D Atkinson, M J Livesey, and D Tulley, *Permutations generated by token passing in graphs*, Theoretical Computer Science **178** (1997), no. 1–2, 103–118.

[AMR02] M D Atkinson, M M Murphy, and N Ruškuc, *Sorting with two ordered stacks in series*, Theor. Comput. Sci. **289** (2002), no. 1, 205–223.

[ARV12] MH Albert, Nik Ruskuc, and Vincent Vatter, *Inflations of geometric grid classes of permutations*, arXiv preprint arXiv:1202.1833 (2012), 1–26.

[AS02] M. D. Atkinson and T. Stitt, *Restricted permutations and the wreath product*, Discrete Mathematics **259** (2002), no. 1–3, 19–36.

[ASV12] M.D. Atkinson, Bruce E. Sagan, and Vincent Vatter, *Counting (3+1)-avoiding permutations*, European Journal of Combinatorics **33** (2012), no. 1, 49–61.

[Atk99] M D Atkinson, *Restricted Permutations*, Discrete Mathematics **195** (1999), no. 1–3, 27–38.

[AV13] Michael H. Albert and Vincent Vatter, *Generating and enumerating 321-avoiding and skew-merged simple permutations*, The Electronic Journal of Combinatorics **20** (2013), 44.

[B0́3] M Bóna, *A survey of stack-sorting disciplines*, Electron. J. Combin **9** (2003), no. 2, 1–16.

[BBL98] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw, *Pattern matching for permutations*, Information Processing Letters **65** (1998), no. 5, 200–209.

[BCdMR08] Anne Bergeron, Cedric Chauve, Fabien de Montgolfier, and Mathieu Raffinot, *Computing Common Intervals of K Permutations, with Applications to Modular Decomposition of Graphs*, SIAM J. Discret. Math. **22** (2008), no. 3, 1022–1039.

[Bev13] David Bevan, *Growth rates of geometric grid classes of permutations*, arXiv preprint arXiv:1306.4246 (2013), 1–8.

[BHV08a] Robert Brignall, Sophie Huczynska, and Vincent Vatter, *Decomposing simple permutations, with enumerative consequences*, Combinatorica **28** (2008), no. 4, 1–14.

[BHV08b] ______, *Simple permutations and algebraic generating functions*, Journal of Combinatorial Theory, Series . . . **115** (2008), no. 3, 423–441.

[BRBP10] Mathilde Bouvel, Dominique Rossin, Frédérique Bassino, and Adeline Pierrot, *Deciding the finiteness of the number of simple permutations contained in a wreath-closed class is polynomial*, ArXiv e-prints (2010), no. 07, 1–14.

[Bri07] Robert Brignall, *Simplicity in Relational Structures and Application to Permutation Classes*, Ph.D. thesis, University of St Andrews, 2007.

[Bri10] ______, *A survey of simple permutations*, Permutation Pattern, vol. 376, London Mathematical Society Lecture Note Series, 2010, pp. 41–65.

[Bri12] ______, *Grid classes and Partial Well Order*, Journal of Combinatorial Theory, Series A **119** (2012), no. 1, 99–116.

[BRV08] Robert Brignall, N Ruškuc, and Vincent Vatter, *Simple permutations: decidability and unavoidable substructures*, Theoretical Computer Science **391** (2008), no. 1, 150–163.

[BRV10] Robert Brignall, Nik Ruškuc, and Vincent Vatter, *Simple Extensions of Combinatorial Structures*, Mathematika **05** (2010), 1–24.

[CS59] N. Chomsky and M.P. Schützenberger, *Computer Programming and Formal Systems*, Studies in Logic and the Foundations of Mathematics, vol. 26, Elsevier, 1959.

[DLM11] M Delgado, S Linton, and J Morais, *Automata, A package on automata, Version 1.13*, http://www.fc.up.pt/cmup/mdelgado/automata, 2011.

[Eld04] Murray Elder, *Pattern avoiding permutations are context-sensitive*, ArXiv Mathematics e-prints (2004), no. March 2005, 1–11.

[GAP15] *GAP – Groups, Algorithms, and Programming, Version 4.7.7*, http://www.gap-system.org, 2015.

[HMU06] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[HV06] Sophie Huczynska and Vincent Vatter, *Grid classes and the Fibonacci dichotomy for restricted permutations*, Journal of Combinatorics **13** (2006), no. 2, 54.

[Kit11] Sergey Kitaev, *Patterns in permutations and words*, Springer Science and Business Media, 2011.

[Knu97] Donald E Knuth, *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[Leh60] D H Lehmer, *Teaching combinatorial tricks to a computer*, Proc. Sympos. Appl. Math., Vol. 10, American Mathematical Society, Providence, R.I., 1960, pp. 179–193.

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

[PR12] Adeline Pierrot and Dominique Rossin, *Simple permutations poset*, ArXiv e-prints (2012), 1–15.

[Sip96] Michael Sipser, *Introduction to the Theory of Computation*, 2nd ed., International Thomson Publishing, 1996.

[SS85] Rodica Simion and Frank W. Schmidt, *Restricted Permutations*, European Journal of Combinatorics **6** (1985), no. 4, 383–406.

[ST93] James H. Schmerl and William T. Trotter, *Critically indecomposable partially ordered sets, graphs, tournaments and other binary relational structures*, Discrete Mathematics **113** (1993), no. 1-3, 191–205.

[SV09] Rebecca Smith and Vincent Vatter, *The enumeration of permutations sortable by pop stacks in parallel*, Information Processing Letters **109** (2009), no. 12, 626–629.

[Tar72] Robert Tarjan, *Sorting Using Networks of Queues and Stacks*, J. ACM **19** (1972), no. 2, 341–346.

[UY00] Takeaki Uno and Mutsunori Yagiura, *Fast Algorithms to Enumerate All Common Intervals of Two Permutations*, Algorithmica **26** (2000), no. 2, 2000.

[Vat11] Vincent Vatter, *Small permutation classes*, Proceedings of the London Mathematical Society **103** (2011), no. 5, 879–921.

[Vat12] ———, *Finding regular insertion encodings for permutation classes*, Journal of Symbolic Computation **47** (2012), no. 3, 259–265.

[VW11a] Vincent Vatter and Steve Waton, *On partial well-order for monotone grid classes of permutations*, Order **28** (2011), no. 2, 193–199.

[VW11b] ———, *On points drawn from a circle*, The Electronic Journal of Combinatorics **18** (2011), 1–10.

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

[Wes95] Julian West, *Generating trees and the Catalan and Schröder numbers*, Discrete Mathematics **146** (1995), 247–262.

[XHP05] BMB Xuan, Michel Habib, and Christophe Paul, *Revisiting T. Uno and M. Yagiuras Algorithm*, Algorithms and Computation (2005), 146–155.