

A polynomial algorithm for deciding the finiteness of the number of simple permutations in permutation classes *

Frédérique Bassino

LIPN UMR 7030, Université Paris 13 and CNRS,
99, avenue J.- B. Clément, 93430 Villetaneuse, France.

Mathilde Bouvel

LaBRI UMR 5800, Université de Bordeaux and CNRS,
351, cours de la Libération, 33405 Talence cedex, France.

Adeline Pierrot

LIAFA UMR 7089, Université Paris Diderot and CNRS,
Case 7014, 75205 Paris cedex 13, France.

Dominique Rossin

LIX UMR 7161, Ecole Polytechnique and CNRS,
91128 Palaiseau, France.

January 6, 2012

Abstract

In this article, we describe an algorithm to determine whether a permutation class \mathcal{C} given by a finite basis B of excluded patterns contains a finite number of simple permutations.

1 Introduction

Since the seminal work of Knuth [18] on the class $Av(231)$ of permutations sortable through one stack, numerous articles have investigated the study of permutation classes in combinatorics. Most of the research done in this field concerns *enumeration* questions on permutation classes (see [10, 14, 17] and their references among many others). Another line of research on permutation classes has been emerging for almost a decade: it is interested in properties or results that are less precise but apply to *families* of permutation classes, that are as wide as possible [3, 4, 5, 12, 13, 19]. This new point of view is not purely combinatorial but instead is intimately linked with algorithms. Indeed,

*This work was completed with the support of the ANR project MAGNUM number ANR BLAN-0204.07

when stating general structural results on families of permutation classes, it is natural to associate to an existential theorem an algorithm that *tests* whether a class given in input falls into the family of classes covered by the theorem, and in this case to *compute* the result whose existence is assessed by the theorem.

Certainly the best illustration of this paradigm that can be found in the literature is the result of Albert and Atkinson [3], stating that every permutation class containing a finite number of simple permutation has a finite basis and an algebraic generating function, and its developements by Brignall *et. al.* in [12, 11, 13]. A possible interpretation of this result is that the simple permutations that are contained in a class somehow determine how structured the class is. Indeed, the algebraicity of the generating function is an echo of a deep structure of the class that appears in the proof of the theorem: the permutations of the class (or rather their decomposition trees) can be described by a context-free grammar.

In this above theorem, as well as in other results obtained in this emerging field [3, 4, 6, 12, 13, 19], it appears that *simple permutations* play a crucial rôle. They can be seen as encapsulating most of the difficulties in the study of permutation classes considered in their generality, both in algorithms and combinatorics.

Our work is about these general results that can be obtained for large families of permutation classes, and is resolutely turned to algorithmic considerations. It takes its root in the theorem of Albert and Atkinson that we already mentionned, and follows its developements in [13] and [8].

In [13], Brignall, Ruškuc and Vatter provide a criterion on a finite basis B for deciding whether a permutation class $\mathcal{C} = Av(B)$ contains a finite number of simple permutations. We have seen from [3] that this is a sufficient condition for the class to be well-structured. This criterion has an algorithmic counterpart in [13]: a decision procedure testing from a finite basis B whether $\mathcal{C} = Av(B)$ contains a finite number of simple permutations. Both in the criterion and in the procedure, the set of *proper pin-permutations* introduced in [13] plays a crucial part. The procedure is based on the construction of automata that accept languages of words on a finite alphabet (that are called *pin words*) that encode permutations that do not belong to the class.

In [8], we perform a detailed study of the class of *pin-permutations*, that contains the proper pin-permutations of [13]. This will allow us to precisely control the pin words corresponding to any given pin-permutation, and to modify the automata construction of [13], and have a new algorithm for deciding whether a permutation class given by a finite basis B contains a finite number of simple permutations. The algorithm we obtain is polynomial w.r.t. the sizes of the patterns in B and simply exponential w.r.t. their number, which is a significant improvement to the first decidability procedure of [13]. More precisely we give a $\mathcal{O}(n^{4k})$ algorithm to decide if a finitely based class of permutations $Av(\pi_1, \pi_2, \dots, \pi_k)$ contains a finite number of simple permutations where $n = \sum |\pi_i|$. This result can be improved to $\mathcal{O}(n \log n)$ for substitution-closed permutation classes [7], that is to say the classes of permutations whose bases contain only simple permutations.

The article is organized as follows. Section 2 is a reminder of previous definitions

and results: permutation patterns, pin-permutations and their pin words, brief description of the characterization and the procedure of [13], decomposition trees, special families of pin-permutations playing an important rôle in our work. Section 3 recalls and refines the interpretation of the pattern containment relation between pin-permutations on their pin words. In Section 4, from the characterization of the decomposition trees of pin-permutations obtained in [8], and refining the ideas used in its proof, we are able to describe, for any pin-permutation π the language of pin words that encode pin-permutations containing π as a pattern. Then, based on this result, Section 5 describes a recursive algorithm that builds, for any pin-permutation π , an automaton that accepts the above language (or rather, for technical reasons that will be explained later, a slight modification of this language). Finally, Section 6 puts together this new automata construction and existing algorithms from [13, 2, 9, 7] to provide an algorithm of reasonable complexity to decide, given a finite basis B , whether the class $\mathcal{C} = Av(B)$ contains a finite number of simple permutations. To conclude, we put this result in the context of previous and future research in Section 7.

2 Preliminaries

We recall in this section a few definitions about permutations, pin representations and pin words. More details can be found in [8, 12, 13].

2.1 Permutation classes

A permutation $\sigma \in S_n$ is a bijective function from $\{1, \dots, n\}$ onto $\{1, \dots, n\}$. We represent a permutation either by a word $\sigma = 2314$ or by its *diagram* (see Figure 1 p.4). A permutation $\pi = \pi_1\pi_2 \dots \pi_k$ is a *pattern* of a permutation $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ and we write $\pi \leq \sigma$ if and only if there exist $1 \leq i_1 < i_2 < \dots < i_k \leq n$ such that π is order isomorphic to $\sigma_{i_1} \dots \sigma_{i_k}$. We also say that σ *involves* or *contains* π . If π is not a pattern of σ we say that σ *avoids* π .

Let B be a finite or infinite antichain -*i.e.* a set of pairwise incomparable elements- of permutations. The permutation class of *basis* B denoted $Av(B)$ is the set of all permutations avoiding every element of B .

A permutation is called *simple* if it contains no interval or block, *i.e.* no mapping from $\{i, \dots, (i+l)\}$ to $\{j, \dots, (j+l)\}$, except the trivial ones corresponding to $l = 0$ or $i = j = 1$ and $l = n - 1$. When the basis B contains only simple permutations the permutation class $Av(B)$ is said to be *wreath-closed*. In [3] wreath-closed classes are defined in a different way but Albert and Atkinson proved that both definitions are equivalent.

In this article, permutation classes are given by their bases. The basis we are interested in are finite otherwise, from [3], the class contains infinitely many simple permutations. Our goal is to check whether such a class contains a finite number of simple permutations, ensuring in this way that its generating function is algebraic [3]. As we shall see in the following, a class of particular permutations, called the pin-permutations,

plays a central role in the decision procedure of this problem. For this reason, we record basic definitions and results related with these pin-permutations.

2.2 Pin representations and pin-permutations

A pin in the plane is a point at integer coordinates. A pin p *separates* - horizontally or vertically - the set of pins P from the set of pins Q if and only if a horizontal - resp. vertical - line drawn across p separates the plane into two parts, one of which contains P and the other one contains Q . A pin sequence is a sequence (p_1, \dots, p_k) of pins in the plane such that no two points lie in the same column or line and for all $i \geq 2$, p_i lies outside the bounding box of $\{p_1, \dots, p_{i-1}\}$ and respects one of the following conditions:

- p_i separates p_{i-1} from $\{p_1, \dots, p_{i-2}\}$.
- p_i is independent from $\{p_1, \dots, p_{i-1}\}$, *i.e.*, it does not separate this set into two non empty sets.

A pin sequence represents a permutation σ if and only if it is order isomorphic to its diagram. We say that a permutation σ is a *pin-permutation* if it can be represented by a pin sequence, which is then called a *pin representation* of σ (see Figure 1). Not all permutations are pin-permutations (see for example the permutation σ of Figure 1).

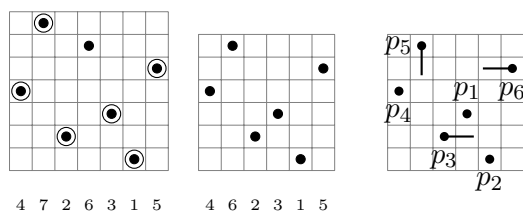


Figure 1: The permutation $\sigma = 4726315$, the pattern $\pi = 462315$ and a pin representation of π .

A *proper* pin representation is a pin representation in which every pin p_i , for $i \geq 3$, separates p_{i-1} from $\{p_1, \dots, p_{i-2}\}$. A *proper* pin-permutation is a permutation that admits a proper pin representation.

Remark 2.1. A pin representation of a simple pin-permutation is always proper as any independent pin p_i with $i \geq 3$ creates a block corresponding to $\{p_1, \dots, p_{i-1}\}$.

2.3 Pin words

Pin representations can be encoded by words on the alphabet $\{1, 2, 3, 4, U, D, L, R\}$ called *pin words*. Consider a pin representation (p_1, \dots, p_n) and choose an origin p_0 in the plane such that (p_0, p_1, \dots, p_n) is a pin sequence. Then every pin p_1, \dots, p_n is encoded by a letter according to the following rules:

- The letter associated with p_i is U -resp. D, L, R - if and only if p_i separates p_{i-1} and $\{p_0, p_1, \dots, p_{i-2}\}$ from the top. -resp. bottom, left, right-.
- The letter associated with p_i is 1 -resp. 2, 3, 4- if and only if p_i is independent from $\{p_0, p_1, \dots, p_{i-1}\}$ and is situated in the up-right -resp. up-left, bottom-left, bottom-right- corner of the bounding box of $\{p_0, p_1, \dots, p_{i-1}\}$.

This encoding is summarized by Figure 2. The region encoded by 1 is called the first *quadrant*. The same goes for 2, 3, 4. The letters L, R, U, D are called *directions*, while 1, 2, 3 and 4 are *numerals*. An important remark is that the definition of pin sequences implies that the pin words do not contain any of the factors $UU, UD, DU, DD, LL, LR, RL$ and RR .

2	U	1
L		R
3	D	4

Figure 2: Encoding of pins by letters.

4R	3R	p_2
41	31	3U
p_1	21	2U

Figure 3: The two letters in each cell indicate the first two letters of the pin word encoding (p_1, \dots, p_n) when p_0 is taken in this cell.

For example, $14L2UR$ (if we place p_0 between p_3 and p_1) and $3DL2UR$ are pin words corresponding to the pin representation of the permutation σ of Figure 1.

To each pin word corresponds a unique pin representation, hence a unique permutation. Because of the choice of the origin p_0 , each pin-permutation of size greater than 1 has at least 6 pin words. A *strict* (resp. *quasi-strict*) pin word is a pin word that begins with a numeral (resp. two numerals) followed only by directions.

Remark 2.2. Notice that strict pin words always encode proper pin representations, whereas a proper pin representation is encoded by both strict and quasi-strict pin words depending on p_0 . Following this idea, another remark is that a pin-permutation is proper if and only if it admits a strict pin word.

The language \mathcal{SP} of strict pin words can be described by the following regular expression:

$$(1 + 2 + 3 + 4) \left((\epsilon + U + D) ((L + R)(U + D))^* + (\epsilon + L + R) ((U + D)(L + R))^* \right).$$

We introduce the usual *shuffle product* on sequences of words that will be used to characterize sets of pin words in the sequel.

Definition 2.3. Let $a = (a_1, a_2, \dots, a_q)$ and $b = (b_1, b_2, \dots, b_r)$ be two sequences. The shuffle product $a \sqcup b$ of a and b is defined as

$$a \sqcup b = \{c = (c_1, \dots, c_{q+r}) \mid \exists I = \{i_1, \dots, i_q\}, J = \{j_1, \dots, j_r\}, I \cap J = \emptyset \\ \text{with } i_1 < \dots < i_q, j_1 < \dots < j_r \text{ and } a = (c_{i_1}, \dots, c_{i_q}), b = (c_{j_1}, \dots, c_{j_r})\}.$$

2.4 Decidability procedure

In [13], Brignall *et al.* studied conditions for a class to contain an infinite number of simple permutations. Introducing three new kinds of permutations they show that this problem is equivalent to looking for an infinite number of permutations of one of these three simpler kinds.

Theorem 2.4. [13] A permutation class $Av(B)$ contains an infinite number of simple permutations if and only if it contains either:

- an infinite number of wedge simple permutations,
- an infinite number of parallel alternations,
- an infinite number of proper pin-permutations.

In Theorem 2.4 above, the proper pin sequences of [13] have been replaced by proper pin-permutations. This is equivalent since each proper pin sequence of size n corresponds to a unique proper pin-permutation of size n , and every proper pin-permutation of size n is associated with at least one and at most 8^n pin sequences of size n .

The definition of the wedge simple permutations and the parallel alternations are not crucial to our work, hence we refer the reader to [13] for more details. What is however important for our purpose is to be able to test whether a class given by a finite basis contains a finite number of parallel alternations or wedge simple permutations.

Alternations and wedge simple permutations, that can be of type 1 or 2, are well characterized in [13]:

Lemma 2.5. [13] The permutation class $Av(B)$ contains only finitely many parallel alternations if and only if B contains an element of every symmetry of the class $Av(123, 2413, 3412)$.

Lemma 2.6. [13] The permutation class $Av(B)$ contains only finitely many wedge simple permutations of type 1 if and only if B contains an element of every symmetry of the class $Av(1243, 1324, 1423, 1432, 2431, 3124, 4123, 4132, 4231, 4312)$.

Lemma 2.7. [13] The permutation class $Av(B)$ contains only finitely many wedge simple permutations of type 2 if and only if B contains an element of every symmetry of the class $Av(2134, 2143, 3124, 3142, 3241, 3412, 4123, 4132, 4231, 4312)$.

Therefore deciding if a class contains an infinite number of wedge simple permutations or parallel alternations is equivalent to checking if elements of its basis involve patterns of size at most 4. Using [2], this test can be made in time $\mathcal{O}(n \log n)$.

In [13] Brignall *et al.* also proved that it is decidable to know if a class contains an infinite number of proper pin-permutations using language theoretic arguments. More precisely they give a transducer which takes as input the pin words of permutations of the basis B of the class and produces as output the set of pin words of permutations containing an element of B as a pattern. Then it remains to determine whether the complement \mathcal{L} of the output language among pin words is finite. Analyzing their procedure, we can prove that it has an exponential complexity due to the resolution of a co-finiteness problem for a regular language given by a non-deterministic automaton.

In the sequel, making use of a characterization of pin-permutations we established in [8], we give a polynomial algorithm for deciding whether a class of permutations contains finitely many proper pin-permutations or not.

2.5 Decomposition trees

Let σ be a permutation of S_n and $\pi^{(1)}, \dots, \pi^{(n)}$ be n permutations of S_{p_1}, \dots, S_{p_n} respectively. Define the *substitution* $\sigma[\pi^{(1)}, \pi^{(2)}, \dots, \pi^{(n)}]$ of $\pi^{(1)}, \pi^{(2)}, \dots, \pi^{(n)}$ in σ (also called *inflation* in [3]) to be the permutation whose graphical representation is obtained from the one of σ by replacing each point σ_i by a block containing the graphical representation of $\pi^{(i)}$. More formally

$$\sigma[\pi^{(1)}, \pi^{(2)}, \dots, \pi^{(n)}] = \mathit{shift}(\pi^{(1)}, \sigma_1) \dots \mathit{shift}(\pi^{(k)}, \sigma_k)$$

where $\mathit{shift}(\pi^{(i)}, \sigma_i) = \mathit{shift}(\pi^{(i)}, \sigma_i)(1) \dots \mathit{shift}(\pi^{(i)}, \sigma_i)(p_i)$ and

$$\mathit{shift}(\pi^{(i)}, \sigma_i)(x) = (\pi^{(i)}(x) + p_{\sigma^{-1}(1)} + \dots + p_{\sigma^{-1}(\sigma_i - 1)}) \text{ for any } x \text{ between } 1 \text{ and } p_i.$$

For example $1\ 3\ 2[2\ 1, 1\ 3\ 2, 1] = 2\ 1\ 4\ 6\ 5\ 3$.

We now introduce decomposition trees. For any $n \geq 2$, let I_n be the permutation $1\ 2 \dots n$ and D_n be $n\ (n-1) \dots 1$. We use the notations \oplus and \ominus for denoting respectively I_n and D_n , for any $n \geq 2$. Notice that in inflations of the form $\oplus[\pi_1, \pi_2, \dots, \pi_n] = I_n[\pi_1, \pi_2, \dots, \pi_n]$ or $\ominus[\pi_1, \pi_2, \dots, \pi_n] = D_n[\pi_1, \pi_2, \dots, \pi_n]$, the integer n is determined without ambiguity by the number of permutations π_i of the inflation.

Definition 2.8. *A permutation σ is \oplus -indecomposable (resp. \ominus -indecomposable) if it cannot be written as $\oplus[\pi_1, \pi_2, \dots, \pi_k]$ (resp. $\ominus[\pi_1, \pi_2, \dots, \pi_k]$), for any $k \geq 2$.*

Theorem 2.9. (first appeared implicitly in [16]) *Every permutation $\sigma \in S_n$ can be uniquely decomposed as either:*

- $\oplus[\pi_1, \pi_2, \dots, \pi_k]$, with $\pi_1, \pi_2, \dots, \pi_k$ \oplus -indecomposable,
- $\ominus[\pi_1, \pi_2, \dots, \pi_k]$, with $\pi_1, \pi_2, \dots, \pi_k$ \ominus -indecomposable,
- $\alpha[\pi_1, \dots, \pi_k]$ with α a simple permutation.

It is important for stating Theorem 2.9 that 12 and 21 are not considered as simple permutations. An equivalent version of this theorem, which includes 12 and 21 among simple permutations, is given in [3]. Notice that the π_i 's correspond to *strong* intervals, *i.e.* intervals that do not overlap any other interval, in the permutation σ , and are necessarily the *maximal* strong intervals of σ strictly included in $\{1, 2, \dots, n\}$. Another important remark is that:

Remark 2.10. *Any block of $\sigma = \alpha[\pi_1, \dots, \pi_k]$ (with α a simple permutation) is either σ itself, or is included in one of the π_i 's.*

Theorem 2.9 can be applied recursively on each π_i leading to a complete decomposition where each permutation is either I_k, D_k (denoted by \oplus, \ominus respectively) or a simple permutation.

Example 2.11. *Let $\sigma = 10\ 13\ 12\ 11\ 14\ 1\ 18\ 19\ 20\ 21\ 17\ 16\ 15\ 4\ 8\ 3\ 2\ 9\ 5\ 6\ 7$. Its recursive decomposition can be written as (see Figure 4)*

$$3\ 1\ 4\ 2[\oplus[1, \ominus[1, 1, 1], 1], 1, \ominus[\oplus[1, 1, 1, 1], 1, 1, 1], 2\ 4\ 1\ 5\ 3[1, 1, \ominus[1, 1], 1, \oplus[1, 1, 1]]].$$

The substitution decomposition recursively applied to maximal strong intervals leads to a tree representation of this decomposition where a substitution $\alpha[\pi_1, \dots, \pi_k]$ is represented by a node labeled α with k ordered children representing the π_i 's. In the sequel we will say the child of a node V instead of the permutation corresponding to the subtree rooted at a child of node V .

Definition 2.12. *The substitution decomposition tree T of the permutation σ is the unique labeled ordered tree encoding the substitution decomposition of σ , where each internal node is either labeled by \oplus, \ominus -those nodes are called linear- or by a simple permutation α -prime nodes-. Each node labeled by α has arity $|\alpha|$ and each node maps onto a strong interval of σ .*

Notice that in substitution decomposition trees, there are no edges between two nodes labeled by \oplus , nor between two nodes labeled by \ominus , since the π_i 's are \oplus -indecomposable (resp. \ominus -indecomposable) in the first (resp. second) item of Theorem 2.9. See Figure 4 for an example.

Theorem 2.13. *[3] Permutations are in one-to-one correspondence with substitution decomposition trees.*

2.6 Oscillations and quasi-oscillations

Among permutations some special ones, called oscillations and quasi-oscillations, play a key role in the characterization of substitution decomposition trees associated with pin-permutations.

Following [13], let us consider the infinite oscillating sequence defined (on $\mathbb{N} \setminus \{0, 2\}$ for regularity of the graphical representation) by $\omega = 4\ 1\ 6\ 3\ 8\ 5\ \dots (2k+2)\ (2k-1)\ \dots$. Figure 5 shows the graphical representation of a prefix of ω .

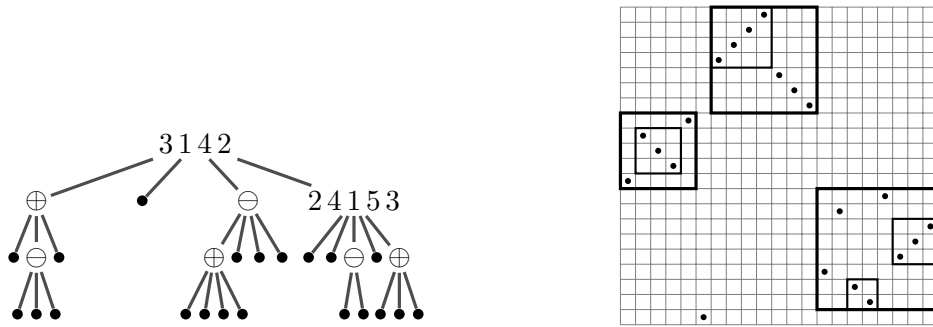


Figure 4: The substitution decomposition tree and the graphical representation (with non-trivial strong intervals marked by rectangles) of permutation $\sigma = 10\ 13\ 12\ 11\ 14\ 1\ 18\ 19\ 20\ 21\ 17\ 16\ 15\ 4\ 8\ 3\ 2\ 9\ 5\ 6\ 7$.

Definition 2.14 (oscillation). An increasing oscillation of size $n \geq 4$ is a simple permutation of size n that is contained as a pattern in ω . The increasing oscillations of smaller size are 1, 21, 231 and 312. A decreasing oscillation is the reverse¹ of an increasing oscillation.

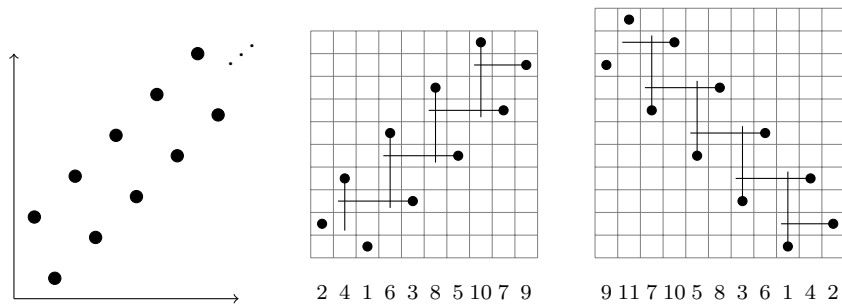


Figure 5: The infinite oscillating sequence, an increasing oscillation of size 10 and a decreasing oscillation of size 11, with a pin representation for each.

As noticed in [8] there are two increasing (resp. decreasing) oscillations of size n for any $n \geq 3$ and permutations 1, 2413 and 3142 are both increasing and decreasing oscillations.

Definition 2.15 (quasi-oscillations [8]). An increasing quasi-oscillation of size $n \geq 6$ is obtained from an increasing oscillation ξ of size $n-1$ by the addition of either a minimal element at the beginning of ξ or a maximal element at the end of ξ , followed by a flip of an element of ξ according to the rules of Table 1. The element that is flipped is called

¹The reverse of $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ is $\sigma^r = \sigma_n \dots \sigma_2\sigma_1$

the outer point of the quasi-oscillation. We also define the auxiliary substitution point to be the point added to ξ , and the main substitution point according to Table 1.²

Element inserted	Pattern formed by $\xi_1\xi_2\xi_3$	Pattern formed by $\xi_{n-3}\xi_{n-2}\xi_{n-1}$	Flipped element which becomes	Main substitution point
max	231	132	left-most	right-most	largest
max	231	312	left-most	right-most	right-most
max	213	132	smallest	largest	largest
max	213	312	smallest	largest	right-most
min	231	132	largest	smallest	left-most
min	231	312	right-most	left-most	left-most
min	213	132	largest	smallest	smallest
min	213	312	right-most	left-most	smallest

Table 1: Flips and main substitution points in increasing quasi-oscillations.

Furthermore, for $n = 4$ or 5 , there are two increasing quasi-oscillations of size n : 2413 , 3142 , 25314 and 41352 . We do not define the outer point of a quasi-oscillation of size less than 6.

Finally, a decreasing quasi-oscillation is the reverse of an increasing quasi-oscillation.

As noticed in [8] there are four increasing (resp. decreasing) quasi-oscillations of size n for any $n \geq 6$, two of size 4 (2413 and 3142) and two of size 5 (25314 and 41352).

Remark 2.16. *Oscillation and quasi-oscillations are simple pin-permutations.*

We refer the reader to [8] for further properties of oscillations and quasi-oscillations.

3 From pattern containment to piecewise factor relation

In this section we show how to transform the pattern containment relation on permutations into a piecewise factor relation between words.

First recall the definition of the partial order \preceq on pin words introduced in [13].

Definition 3.1. *Let u and w be two pin words. We decompose u in terms of its strong numeral-led factors as $u = u^{(1)} \dots u^{(j)}$, a strong numeral-led factor being a strict pin word. We then write $u \preceq w$ if w can be chopped into a sequence of factors $w = v^{(1)}w^{(1)} \dots v^{(j)}w^{(j)}v^{(j+1)}$ such that for all $i \in \{1, \dots, j\}$:*

- if $w^{(i)}$ begins with a numeral then $w^{(i)} = u^{(i)}$, and
- if $w^{(i)}$ begins with a direction, then $v^{(i)}$ is nonempty, the first letter of $w^{(i)}$ corresponds to a point lying in the quadrant specified by the first letter of $u^{(i)}$, and all other letters in $u^{(i)}$ and $w^{(i)}$ agree.

²The first line of Table 1 reads as: If a maximal element is added to ξ , ξ starts (resp. ends) with a pattern 231 (resp. 132), then the corresponding increasing quasi-oscillation β is obtained by flipping the left-most point of ξ to the right-most (in β), and the main substitution point is the largest point of ξ .

Notice that with the same notations, $u \preceq w$ if and only if $u^{(i)} \preceq v^{(i)}w^{(i)}$ for all $i \in \{1, \dots, j\}$.

This order is closely related to the pattern containment order \leq on permutations.

Lemma 3.2. [13] *If the pin word w corresponds to the permutation σ and $\pi \leq \sigma$ then there is a pin word u corresponding to π with $u \preceq w$. Conversely if $u \preceq w$ then the permutation corresponding to u is contained in the permutation corresponding to w .*

In what follows, σ is a proper pin-permutation. So we can choose a strict pin word w that encodes σ (see Remark 2.2). As a consequence of Lemma 3.2, checking whether a permutation π is a pattern of σ is equivalent to checking whether there exists a pin word u corresponding to π with $u \preceq w$. The relation $u \preceq w$ on pin words is nearly a piecewise factor relation, the factors being determined by the strong numeral led factors of u . We use an encoding of pin words that maps the relation \preceq on an actual piecewise factor relation that we introduced in [7] and recall hereafter. For our purpose, it is enough to consider the case where w is a strict pin word since every proper pin-permutation can be encoded by a strict pin word.

Definition 3.3. *Let \mathcal{M} be the set of words of length greater than or equal to 3 over the alphabet L, R, U, D such that R, L is followed by U, D and conversely.*

We define a bijection ϕ from \mathcal{SP} to \mathcal{M} as follows. For any strict pin word $u \in \mathcal{SP}$ such that $u = u'u''$ with $|u'| = 2$, we set $\phi(u) = \varphi(u')u''$ where φ is given by:

$1R \mapsto RUR$	$2R \mapsto LUR$	$3R \mapsto LDR$	$4R \mapsto RDR$
$1L \mapsto RUL$	$2L \mapsto LUL$	$3L \mapsto LDL$	$4L \mapsto RDL$
$1U \mapsto URU$	$2U \mapsto ULU$	$3U \mapsto DLU$	$4U \mapsto DRU$
$1D \mapsto URD$	$2D \mapsto ULD$	$3D \mapsto DLD$	$4D \mapsto DRD$

For any $n \geq 2$, the map ϕ is a bijection from the set \mathcal{SP}_n of strict pin words of length n to the set \mathcal{M}_{n+1} of words of \mathcal{M} of length $n + 1$. Furthermore, it satisfies, for any $u \in \mathcal{SP}$, $u_i = \phi(u)_{i+1}$ for any $i \geq 2$.

In the above table, we can notice that, for any $u \in \mathcal{SP}$, the first two letters of $\phi(u)$ are sufficient to determine the first letter of u (which is a numeral). Thus it is natural to extend the definition of ϕ to words of length 1 (that do not belong to \mathcal{SP} by definition) by setting $\phi(1) = \{UR, RU\}$, $\phi(2) = \{UL, LU\}$, $\phi(3) = \{DL, LD\}$ and $\phi(4) = \{RD, DR\}$, and by defining consistently $\phi^{-1}(v) \in \{1, 2, 3, 4\}$ for any v in $\{LU, LD, RU, RD, UL, UR, DL, DR\}$.

Notice that our bijection consists of replacing the only numeral in any strict pin word by two directions. Lemma 3.4 below shows that for each strict pin word w , we know in which quadrant lies every pin of the pin representation corresponding to w .

Lemma 3.4. [7] *Let w be a strict pin word and p the pin representation corresponding to w . For any $i \geq 2$, set*

$$q(w_{i-1}, w_i) = \begin{cases} \phi^{-1}(w_{i-1}w_i) & \text{if } i \geq 3 \\ \phi^{-1}(bc) & \text{if } i = 2 \text{ and } \phi(w_1w_2) = abc \end{cases}$$

Then for any $i \geq 2$, $q(w_{i-1}, w_i)$ is a numeral indicating the quadrant in which p_i lies with respect to $\{p_0, \dots, p_{i-2}\}$.

By definition, strong numeral led factors of any pin word u are strict pin words. Therefore we first study how the relation $u \preceq w$ is mapped on $\phi(u), \phi(w)$ when u is a strict pin word.

We first consider pin words of length greater or equal to 2.

Lemma 3.5. [7] *For any strict pin words u and w , such that $|u| \geq 2$, $u \preceq w$ if and only if $\phi(u)$ is a factor of $\phi(w)$.*

If $|u| = 1$ then $\phi(u)$ contains two words and we can prove in a similar way the following lemma.

Lemma 3.6. *For any strict pin words u and w such that $|u| = 1$, $u \preceq w$ if and only if $\phi(w)$ has a factor in $\phi(u)$.*

Remark 3.7. *Note that the relation $u \preceq w$ is still defined when $w \in \mathcal{M}$. Moreover the map ϕ can be extended to words of \mathcal{M} as the identity map. This allows us to generalize Lemmas 3.5 and 3.6 to the case where w belongs to \mathcal{M} .*

When the pin word u is not strict, the two previous lemmas can be extended formalizing the idea of piecewise factors as explained at the beginning of this section. Let u be a pin word and $u = u^{(1)} \dots u^{(j)}$ its decomposition in terms of its strong numeral-led factors. We set

$$\mathcal{L}(u) = A^* \phi(u^{(1)}) A^* \phi(u^{(2)}) \dots A^* \phi(u^{(j)}) A^* \cap \mathcal{M}$$

where $A = \{L, R, U, D\}$.

Theorem 3.8. *Let σ be a proper pin-permutation, π a permutation and w a strict pin word associated to σ . Then $\pi \leq \sigma$ if and only if there exists a pin word u associated to π such that $\phi(w) \in \mathcal{L}(u)$.*

Proof. This result is a generalization of Lemmas 3.5 and 3.6. Note that, by definition, $\phi(w)$ belongs to \mathcal{M} .

Assume that $\pi \leq \sigma$, then π is a pin-permutation, and from Lemma 3.2 there exists a pinword u encoding π such that $u \preceq w$. We decompose u into strong numeral-led factors $u = u^{(1)} u^{(2)} \dots u^{(j)}$. Recall that by definition all $u^{(i)}$ are strict pin words. Then w can be decomposed into $w = v^{(1)} w^{(1)} v^{(2)} w^{(2)} \dots w^{(j)} v^{(j+1)}$ and for all $i \in \{1, \dots, j\}$, $u^{(i)} \preceq v^{(i)} w^{(i)}$. As w is a strict pin word, $v^{(1)} w^{(1)}$ is a strict pin word and, for all $i \geq 2$, $v^{(i)} w^{(i)}$ belong to \mathcal{M} . Lemmas 3.5 and 3.6 insure that for all $i \in \{1, \dots, j\}$, $\phi(v^{(i)} w^{(i)})$ contains a factor equal to (or belonging to) $\phi(u^{(i)})$. Finally since w is a strict pinword, $\phi(w) = \phi(v^{(1)} w^{(1)}) \phi(v^{(2)} w^{(2)}) \dots \phi(v^{(j)} w^{(j)}) \phi(v^{(j+1)})$, and therefore the factors $\phi(u^{(1)}) \dots \phi(u^{(j)})$ appear in $\phi(w)$, in this order from left to right, being disjoint. In other words $\phi(w) \in \mathcal{L}(u)$.

Conversely if there exists a pinword u encoding π such that $\phi(w) \in \mathcal{L}(u)$, $\phi(w)$ can be decomposed into $t^{(1)} \dots t^{(j+1)}$, with $t^{(i)} \in A^* \phi(u^{(i)}) \cap \mathcal{M}$ for $i \in \{1, \dots, j\}$ and

$t^{(j+1)} \in \mathcal{M}$. For $i \in \{2, \dots, j\}$, $\phi(u^{(i)})$ is a factor of $t^{(i)} = \phi(t^{(i)})$ and, by Lemmas 3.5 and 3.6, we get $u^{(i)} \preceq t^{(i)}$. For $i = 1$, define t to be the unique strict pin word such that t_1 is equal (or belongs) to $\phi(t)$. Then $\phi(u^{(1)})$ is a factor of $\phi(t)$ and, by Lemmas 3.5 and 3.6, we get $u^{(1)} \preceq t$. Since w is a strict pin word, $w = tt^{(2)} \dots t^{(j+1)}$, so that $u \preceq w$. Finally from Lemma 3.2, we conclude that $\pi \leq \sigma$. \square

4 Pin words of pin-permutations

In this section, our goal is to describe the set $P(\sigma)$ of pin words associated to a pin-permutation σ . In [8], it is proved that pin-permutations can be characterized by their decomposition trees. The key idea of this section is to follow the decomposition established in [8]. Recall that the set \mathcal{S} of substitution decomposition trees of pin-permutations is recursively characterized as follows:

$$\begin{aligned}
\mathcal{S} = & \bullet + \begin{array}{c} \oplus \\ \diagup \quad \diagdown \\ \mathcal{E}^+ \quad \mathcal{E}^+ \quad \dots \quad \mathcal{E}^+ \end{array} + \begin{array}{c} \oplus \\ \diagup \quad \dots \quad \diagdown \\ \mathcal{E}^+ \quad \dots \quad \mathcal{E}^+ \\ \triangle \\ \mathcal{N}^+ \end{array} + \begin{array}{c} \ominus \\ \diagup \quad \diagdown \\ \mathcal{E}^- \quad \mathcal{E}^- \quad \dots \quad \mathcal{E}^- \end{array} \\
& + \begin{array}{c} \ominus \\ \diagup \quad \dots \quad \diagdown \\ \mathcal{E}^- \quad \dots \quad \mathcal{E}^- \\ \triangle \\ \mathcal{N}^- \end{array} + \begin{array}{c} \alpha \\ \diagup \quad \dots \quad \diagdown \\ \bullet \quad \bullet \quad \dots \quad \bullet \end{array} + \begin{array}{c} \alpha \\ \diagup \quad \dots \quad \diagdown \\ \bullet \quad \bullet \quad \dots \quad \bullet \\ \triangle \\ \mathcal{S} \setminus \{\bullet\} \end{array} \\
& + \begin{array}{c} \beta^+ \\ \diagup \quad \dots \quad \diagdown \\ \bullet \quad \bullet \quad \dots \quad \bullet \\ \triangle \\ \mathcal{S} \setminus \{\bullet\} \end{array} + \begin{array}{c} \beta^- \\ \diagup \quad \dots \quad \diagdown \\ \bullet \quad \bullet \quad \dots \quad \bullet \\ \triangle \\ \mathcal{S} \setminus \{\bullet\} \end{array}
\end{aligned}$$

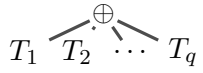
where \mathcal{E}^+ (resp. \mathcal{E}^-) is the set of decomposition trees of increasing (resp. decreasing) oscillations, \mathcal{N}^+ (resp. \mathcal{N}^-) the set of decomposition trees of pin-permutations that are not increasing (resp. decreasing) oscillations and whose root is not \oplus (resp. \ominus), α any simple pin-permutation and β^+ (resp. β^-) any increasing (resp. decreasing) quasi-oscillation. In this definition only terms which contain a subtree labeled by \mathcal{N}^+ , \mathcal{N}^- or \mathcal{S} are recursive.

Our proof is naturally divided into several parts. First, we study the non-recursive cases, then the recursive case with a linear root and finally the recursive case with a prime root. In each case, we give the rational language of pin words associated to the studied permutations. We start with a preliminary study of the ways blocks of decomposition trees with linear root can be read in a pin representation. These first results will be useful both in the non-recursive and the recursive case.

4.1 Reading of blocks

Definition 4.1. *Let σ be a pin-permutation and $p = (p_1, \dots, p_n)$ be a pin representation of σ . For any set D of points of σ , if k is the number of maximal factors $p_i, p_{i+1}, \dots, p_{i+k}$*

of p that contain only points of D , we say that D is read in k times by p . If C is another set of points of σ , we say that D is read before (resp. read entirely before) C if the first pin in C appears in p after the first pin belonging to D (resp. after all pins belonging to D).

Let σ be a permutation whose decomposition tree has a linear root, wlog assume that $T_\sigma =$  and let $p = (p_1, \dots, p_n)$ one of its pin representations. In

the sequel, we denote by i_0 the index of the child which contains p_1 .

Lemma 4.2. *Let $1 \leq i, j \leq q$ such that either $i < j < i_0$ or $i_0 < j < i$. Then T_j is read entirely before T_i .*


Proof. Let $\ell = \min\{\ell', p_{\ell'} \in T_i\}$. Let \mathcal{B} be the bounding box of $\{p_1, \dots, p_\ell\}$. As $p_1 \in T_{i_0}$, $T_j \subset \mathcal{B}$, (see Figure 6) hence it is entirely read in p before p_ℓ . \square

The previous lemma gives the possible orders in which blocks are read. Now we characterize blocks T_i which can be read in several times. When this is the case, we will prove that the decomposition tree is of a specific shape. This can indeed be deduced from the following lemmas:


Lemma 4.3. *For every $k \in \{1, \dots, n\}$ there is at most one block whose reading has started and is not finished after (p_1, p_2, \dots, p_k) .*

Proof. Without loss of generality we consider that the root of the decomposition tree is labeled with \oplus as shown in Figure 6. Suppose that pins $\{p_1, \dots, p_k\}$ have already been read. By Lemma 4.2 there exists at most one block T_i with $i < i_0$ and at most one block T_m with $m > i_0$ whose readings have started and are not finished. Note that

$$i = \min\{\ell \mid \exists h \in \{1, \dots, k\}, p_h \in T_\ell\}.$$

The same goes for m changing the minimum into a maximum. If the reading of T_i is not finished, since T_i is \oplus -indecomposable, there must exist a pin p_q in zone  (see Figure 7). Such a pin is on the side of the bounding box, and the same remark goes for the block T_m . But from Lemma 2.17 of [8] there is at most one pin lying on the sides of a bounding box, thus for every $k \in \{1 \dots n\}$ there is at most one block whose reading has started and is not finished after (p_1, p_2, \dots, p_k) . \square

Lemma 4.4. *Each block T_i is read in one time by p , except perhaps T_{i_0} .*

Proof. Consider a block T_i with $i \neq i_0$ which is read in more than one time by p . Consider the pin p_{k+1} which is the first pin outside T_i after p has started reading T_i . As p_1 is in T_{i_0} , p_1 is outside T_i and the bounding box of $\{p_1, p_2, \dots, p_{k-1}, p_k\}$ allows to define a zone  as shown in Figure 7. Since T_i is \oplus -indecomposable, there is at least 1 pin in this zone. This pin is on the side of the bounding box of $\{p_1, p_2, \dots, p_{k-1}\}$ so it is p_k by Lemma 2.17 of [8], so $p_k \in T_i$ which is absurd. \square

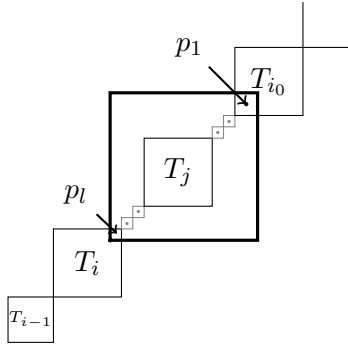


Figure 6: Proof of Lemma 4.2.

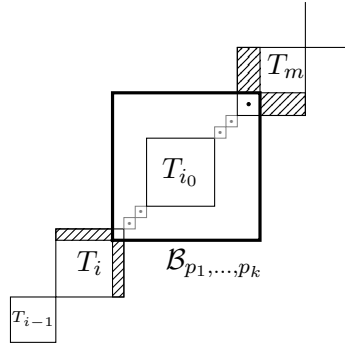


Figure 7: Proof of Lemma 4.3.

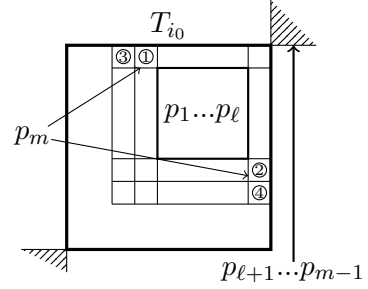


Figure 8: T_{i_0} is read in two times.

Lemmas 4.3 and 4.4 have the consequences summarized hereafter:

Remark 4.5. If a pin representation p reads a block T in several times, then $p_1 \in T$. If T_{i_0} is read in several times, say p_1, \dots, p_ℓ then p_k, \dots with $k > \ell + 1$, each of the pins $(p_j)_{\ell < j < k}$ forms a maximal strong interval otherwise reading p_j would leave the current block unfinished like T_{i_0} .

If T_{i_0} can be read in several times, this means that the decomposition tree of the whole permutation has a special shape given in the following lemma.

Lemma 4.6. The only permutations σ whose decomposition trees have a root \oplus in which a block T_{i_0} can be read in several times are those whose decomposition trees have one of the following shapes where ξ^+ is an increasing oscillation of size at least 4:

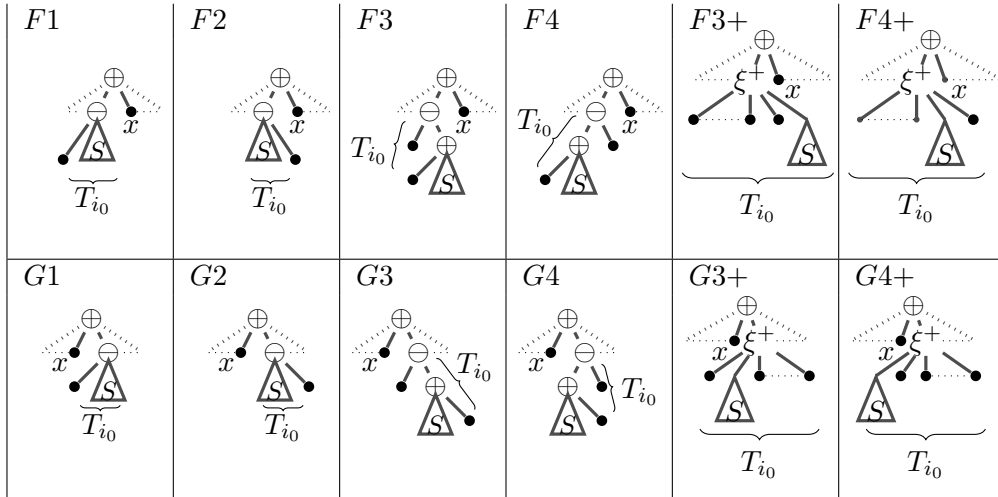


Figure 9: Decomposition tree when T_{i_0} can be read in several times.

Moreover, for these permutations, if T_{i_0} is read in more than one time, then T_{i_0} is

read in two times, the first part being S , the second one being the remaining leaves of T_{i_0} with only the point x read in between.

Proof. Let $p = (p_1, p_2, \dots, p_n)$ be a pin representation in which T_{i_0} is read in several times, the first part being p_1, \dots, p_ℓ . Then $p_{\ell+1}, \dots, p_{m-1}$ belong to other blocks until $p_m \in T_{i_0}$.

As T_{i_0} is a child of the root \oplus , each pin p_i with $i \in \{\ell + 1, \ell + 2, \dots, m - 1\}$ lies in one of the zones $\text{\textbackslash}\text{\textbackslash}\text{\textbackslash}$ as shown in Figure 8. But if both zones contain at least one pin the bounding box of $\{p_1, \dots, p_{m-1}\}$ would contain T_{i_0} and thus p_m cannot respect the externality condition. Hence all pins p_i with $i \in \{\ell + 1, \ell + 2, \dots, m - 1\}$ are in the same zone.

Assume for example that $\{p_{\ell+1}, \dots, p_{m-1}\}$ are in the upper right zone of Figure 8. If p_m respects the independence condition, it must lie in the lower left corner of the bounding box of $\{p_1, \dots, p_\ell\}$ and every future pin of T_{i_0} lies in the same corner leading to a \oplus -decomposable block T_{i_0} which is forbidden. Thus p_m must be a separating pin and $m = \ell + 2$.

As only one point can lie on the sides of a bounding box, the distance between the bounding box of $\{p_1, \dots, p_\ell\}$ and $\{p_{\ell+1}\}$ is 1 leaving only four possible positions for p_m as depicted in Figure 8. If p_m is at distance 1 of the bounding box of $\{p_1, \dots, p_\ell\}$ either in position ① (case $F1$ on Figure 10) or ② (case $F2$) then pins $\{p_1, p_2, \dots, p_\ell, p_m\}$ form an interval and thus represent T_{i_0} (because T_{i_0} is \oplus -indecomposable). Otherwise p_m lies at distance 2 of the bounding box of $\{p_1, \dots, p_\ell\}$ either in position ③ (case $F3$ and $F3+$) or ④ (case $F4$ and $F4+$). Suppose that it is in position ④ then p_{m+1} is a left pin separating p_m from the preceding ones. It has again two different positions: the first one being at distance 1 from the preceding bounding box, thus ending T_{i_0} (case $F4$), the second being at distance 2 (case $F4+$). This process can be repeated alternating between left and down pins until one of them, say p_{m+k} is at distance one of the current bounding box, ending the block T_{i_0} .

Thus we have proved that T_{i_0} is read in exactly two times. It is then straightforward to check that these permutations are exactly those described by their decomposition tree in the statement of the theorem. \square

Note that in the proof of the preceding lemma the order in which points corresponding to the leaves of $T_{i_0} \setminus S$ are read is uniquely determined, leading to the following remark:

Remark 4.7. *In Lemma 4.6, if T_{i_0} is read in two times with the first part fixed, then the points in the second part and their order are uniquely determined.*

4.2 Non-recursive cases

Permutation of size 1. Notice first that the permutation $\sigma = 1$ whose decomposition tree is a leaf has exactly four pin words $-1, 2, 3, 4-$.

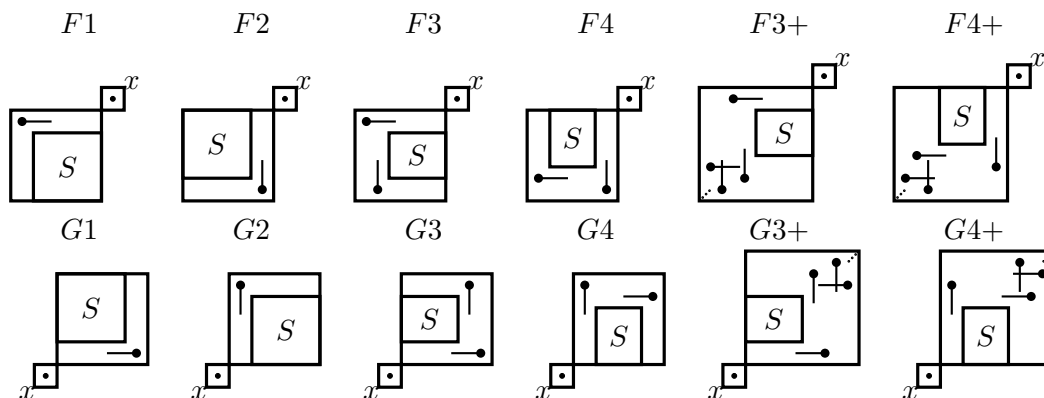


Figure 10: Graphical representation when T_{i_0} can be read in two times.

Simple permutations. The only pin-permutations whose decomposition tree has a prime root and is non-recursive are the simple pin-permutations. Moreover all pin representations of a simple pin-permutation σ are proper and, by [8, Lemma 4.3], at most four pairs of pins can appear at the beginning of such a representation. As for each pair of such pins there are at most 16 pin words associated, a simple pin-permutation has at most 64 pin words. An efficient method to compute them is given in Section 5.2 (p.28).

Permutations whose decomposition tree has a linear root. Without loss of generality we only deal with permutations whose decomposition tree has a root labeled by \oplus . Thus $\sigma = \oplus[\xi_1, \xi_2, \dots, \xi_q]$ where ξ_i are increasing oscillations. The following lemma is then a direct consequence of Lemmas 4.4 and 4.6.

Lemma 4.8. *Let $p = (p_1, p_2, \dots, p_n)$ be a pin representation of σ . The only block ξ_i which can be read in several times is the block ξ_{i_0} to which p_1 belongs. Moreover if p reads ξ_{i_0} in several times, it is read in two times, the second block ξ_i read by p is either ξ_{i_0-1} or ξ_{i_0+1} and is a leaf, denoted x . Moreover the set $E = \xi_{i_0} \cup \{x\}$ is read in one time by p .*

Lemmas 4.2 and 4.8 lead to the following.

Consequence 4.9. *Every pin representation p of σ begins by entirely reading two consecutive children of the root, say ξ_j and ξ_{j+1} , then p reads in one time each of the others ξ_i . Moreover the blocks $\xi_i, i < j$ are read in decreasing order ($\xi_{j-1}, \xi_{j-2}, \dots, \xi_1$) and the blocks $\xi_i, i > j + 1$ are read in increasing order ($\xi_{j+2}, \xi_{j+3}, \dots, \xi_q$).*

Consequence 4.9 implies that the restriction of p to each block ξ_i where $i < j$ (resp. $i > j + 1$) is a pin representation of ξ_i whose origin lies in quadrant 1 (resp. 3) with respect to the bounding box of the set of points of ξ_i . Therefore we introduce the following functions $P^{(\ell)}$: for any increasing oscillation ξ , we denote by $P^{(\ell)}(\xi)$ the set of pin words that represents ξ and whose origin lies in quadrant $\ell = 1$ or 3 with respect

to the points of ξ . Recall that an *active knight* of a permutation is a pair of pins that is a possible start of a pin representation of the permutation. From [8, Lemma 4.3] an oscillation of size at least 5 has exactly two active knights and these active knights are located at both ends of the main diagonal. When the oscillation is increasing, the two points of an active knight are in relative order 21. It is either in horizontal H position $\begin{array}{|c|c|} \hline \square & \square \\ \hline \end{array}$ or in vertical V position $\begin{array}{|c|} \hline \square \\ \hline \end{array}$. Therefore an oscillation has four different types of knights (x, y) with $x, y \in \{H, V\}$. For an increasing oscillation x is the type of the lower left knight and y for the upper right. Note that an even size oscillation has type (H, H) or (V, V) and an odd size one (H, V) or (V, H) . An exhaustive study of the different cases given in Figure 11 leads to the following statement.

Lemma 4.10. *Let ξ be an increasing oscillation of size $n \geq 5$.*

If n is even, let $n = 2p + 2$, then

$$P^{(1)}(\xi) = \begin{cases} 3L(DL)^p & \text{if } \xi \text{ has type } (H, H) \\ 3D(LD)^p & \text{if } \xi \text{ has type } (V, V) \end{cases} \quad P^{(3)}(\xi) = \begin{cases} 1R(UR)^p & \text{if } \xi \text{ has type } (H, H) \\ 1U(RU)^p & \text{if } \xi \text{ has type } (V, V). \end{cases}$$

If n is odd, let $n = 2p + 1$, then

$$P^{(1)}(\xi) = \begin{cases} 3(DL)^p & \text{if } \xi \text{ has type } (H, V) \\ 3(LD)^p & \text{if } \xi \text{ has type } (V, H) \end{cases} \quad P^{(3)}(\xi) = \begin{cases} 1(RU)^p & \text{if } \xi \text{ has type } (H, V) \\ 1(UR)^p & \text{if } \xi \text{ has type } (V, H). \end{cases}$$

For the increasing oscillations of size less than 4, the values of $P^{(1)}$ and $P^{(3)}$ are:

$$\begin{array}{llll} P^{(1)}(1) = 3 & P^{(3)}(1) = 1 & P^{(1)}(21) = \{3D, 3L\} & P^{(3)}(21) = \{1R, 1U\} \\ P^{(1)}(231) = 3DL & P^{(3)}(231) = 1RU & P^{(1)}(312) = 3LD & P^{(3)}(312) = 1UR \\ P^{(1)}(2413) = 3LDL & P^{(3)}(2413) = 3RUR & P^{(1)}(3142) = 3DLD & P^{(3)}(3142) = 1URU \end{array}$$

Remark 4.11. *If the oscillation ξ is of size 2 then $P^{(\ell)}(\xi)$ contains two words, otherwise it is a singleton.*

We are further interested in describing the set of pin words of ξ (denoted $P(\xi)$) and the set of pin words of $\oplus[\xi_\ell, \xi_r]$. This is achieved in the two following lemmas, whose proofs proceed by exhaustive examination of the different cases illustrated in Figure 11.

Lemma 4.12. *Set $Q = \{12, 14, 22, 24, 32, 34, 42, 44\}$, $S_H = \{1R, 2R, 3L, 4L\}$ and $S_V = \{1U, 2D, 3D, 4U\}$. Notice that Q (resp. S_H , resp. S_V) is the set of pin words of the permutation 21 that are quasi-strict (resp. that are strict and end with R or L , resp. with U or D).*

Let ξ be an increasing oscillation of size $n \geq 5$.

If n is even, let $n = 2p + 2$, then

$$P(\xi) = \begin{cases} (Q + S_H) \cdot (DL)^p + (Q + S_H) \cdot (UR)^p & \text{if } \xi \text{ has type } (H, H) \\ (Q + S_V) \cdot (LD)^p + (Q + S_V) \cdot (RU)^p & \text{if } \xi \text{ has type } (V, V). \end{cases}$$

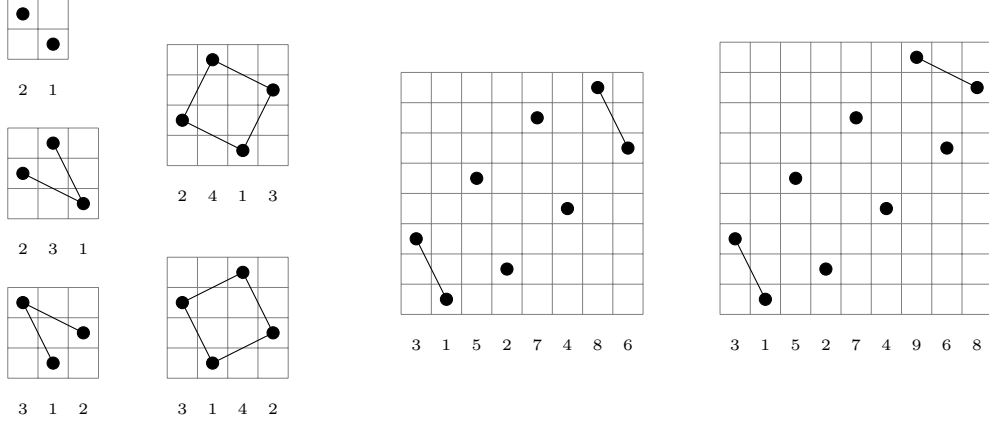


Figure 11: The increasing oscillations of size less than 5 and two oscillations respectively of size 8 with type (V, V) and 9 with type (V, H) in which active knights are marked.

If n is odd, let $n = 2p + 1$, then

$$P(\xi) = \begin{cases} (Q + S_V) \cdot L(DL)^{p-1} + (Q + S_H) \cdot U(RU)^{p-1} & \text{if } \xi \text{ has type } (H, V) \\ (Q + S_H) \cdot D(LD)^{p-1} + (Q + S_V) \cdot R(UR)^{p-1} & \text{if } \xi \text{ has type } (V, H). \end{cases}$$

For the increasing oscillations of size less than 5, we have:

$$\begin{aligned} P(1) &= \{1, 2, 3, 4\} & P(21) &= Q + S_H + S_V \\ P(231) &= (Q + S_H) \cdot U + (Q + S_V) \cdot L & P(312) &= (Q + S_H) \cdot D + (Q + S_V) \cdot R \\ P(2413) &= (Q + S_H) \cdot (UR + DL) + (Q + S_V) \cdot (RD + LU) \\ P(3142) &= (Q + S_H) \cdot (UL + DR) + (Q + S_V) \cdot (RU + LD) \end{aligned}$$

In particular, $|P(\xi)| \leq 48$ for any oscillation ξ .

Definition 4.13. For any pair of increasing oscillations (ξ_ℓ, ξ_r) , we denote by

- $P^{\text{double}}(\xi_\ell, \xi_r)$ the set of pin words encoding a pin representation of $\oplus[\xi_\ell, \xi_r]$.
- $P^{\text{mix}}(\xi_\ell, \xi_r)$ the set of pin words encoding a pin representation of $\oplus[\xi_\ell, \xi_r]$ that reads one of the two oscillations in two times.

Lemma 4.14. Let ξ_ℓ and ξ_r be two increasing oscillations.

If none of these two oscillations is of size 1, or if both of them are of size 1, then $P^{\text{mix}}(\xi_\ell, \xi_r)$ is empty.

Otherwise, assume w.l.o.g that $\xi_\ell = 1$, and denote by $n = 2p + q$ the size of ξ_r , $q \in \{0, 1\}$. If $n > 2$, then

$$P^{\text{mix}}(\xi_\ell, \xi_r) = \begin{cases} (13 + 23 + 33 + 43 + 1D + 4D) \cdot (RU)^{p-1} R^q & \text{if } \xi \text{ has type } (H, H) \text{ or } (H, V) \\ (13 + 23 + 33 + 43 + 1L + 2L) \cdot (UR)^{p-1} U^q & \text{if } \xi \text{ has type } (V, V) \text{ or } (V, H). \end{cases}$$

In the case $|\xi_r| = 2$, i.e. $\xi_r = 21$, we have

$$P^{mix}(\xi_\ell, \xi_r) = (13 + 23 + 33 + 43 + 1D + 4D) \cdot R + (13 + 23 + 33 + 43 + 1L + 2L) \cdot U.$$

In particular, $P^{mix}(\xi_\ell, \xi_r) \leq 12$ for any oscillations ξ_ℓ, ξ_r .

The above expressions of P, P^{mix} and $P^{(\ell)}$ allow us to give the following explicit expression of P^{double} :

Lemma 4.15. *For any pair of increasing oscillations (ξ_ℓ, ξ_r) ,*

$$P^{double}(\xi_\ell, \xi_r) = P(\xi_r) \cdot P^{(1)}(\xi_\ell) \sqcup P(\xi_\ell) \cdot P^{(3)}(\xi_r) \sqcup P^{mix}(\xi_\ell, \xi_r).$$

Using the shuffle product defined p.5 and Consequence 4.9, we completely characterize pin words that encode a permutation $\sigma = \oplus[\xi_1, \xi_2, \dots, \xi_q]$ where every ξ_i is an increasing oscillation:

Theorem 4.16. *The set $P(\sigma)$ of pin words of a permutation $\sigma = \oplus[\xi_1, \dots, \xi_q]$ where every ξ_i is an increasing oscillation is:*

$$P(\sigma) = \bigcup_{1 \leq j \leq q-1} P^{double}(\xi_j, \xi_{j+1}) \cdot \left((P^{(1)}(\xi_{j-1}), \dots, P^{(1)}(\xi_1)) \sqcup (P^{(3)}(\xi_{j+2}), \dots, P^{(3)}(\xi_q)) \right).$$

Lemmas 4.10 and 4.15 give explicit expressions for $P^{(1)}(\xi)$, $P^{(3)}(\xi)$ and $P^{double}(\xi_\ell, \xi_r)$ for every increasing oscillations ξ , ξ_ℓ and ξ_r , hence an explicit expression for $P(\sigma)$. Note that these results hold for increasing oscillations and similar ones can be obtained for decreasing oscillations. In Section 6 it will be shown how this set of pin words can be recognized in polynomial time by a deterministic finite automaton.

4.3 Recursive case: decomposition trees with a linear root

In this section we focus on permutations whose decomposition tree has a linear root and a child T_{i_0} which is not an increasing oscillation. From [8, Lemma 3.3] T_{i_0} is then the first child read. Lemma 4.6 gives a characterization of permutations in which T_{i_0} can be read in several times. Moreover from Remark 4.7 if T_{i_0} is read in two times the first part S being fixed, then the order of the points of the remaining part is uniquely determined. Nevertheless, since some permutations can satisfy several conditions $F1$ to $G4+$ of Lemma 4.6, the first part S to be read is not uniquely determined. For example every permutation satisfying $F3$ also satisfies $F1$, and some permutations satisfy both $F1$ and $G2$. In Figure 12 we classify the permutations according to the conditions they satisfy.

Let (H) be the set of permutations in which T_{i_0} can be read in several times. Then any permutation of (H) satisfies one of the conditions of Figure 12. One can check by an exhaustive verification that there is no other combination (up to symmetry) of the conditions $F1$ to $G4+$ of Lemma 4.6. Moreover as T_{i_0} is not an increasing oscillation, $|S| \geq 2$ and $|T| \geq 1$.

Recall that $P(\sigma)$ denotes the set of pin words encoding σ .

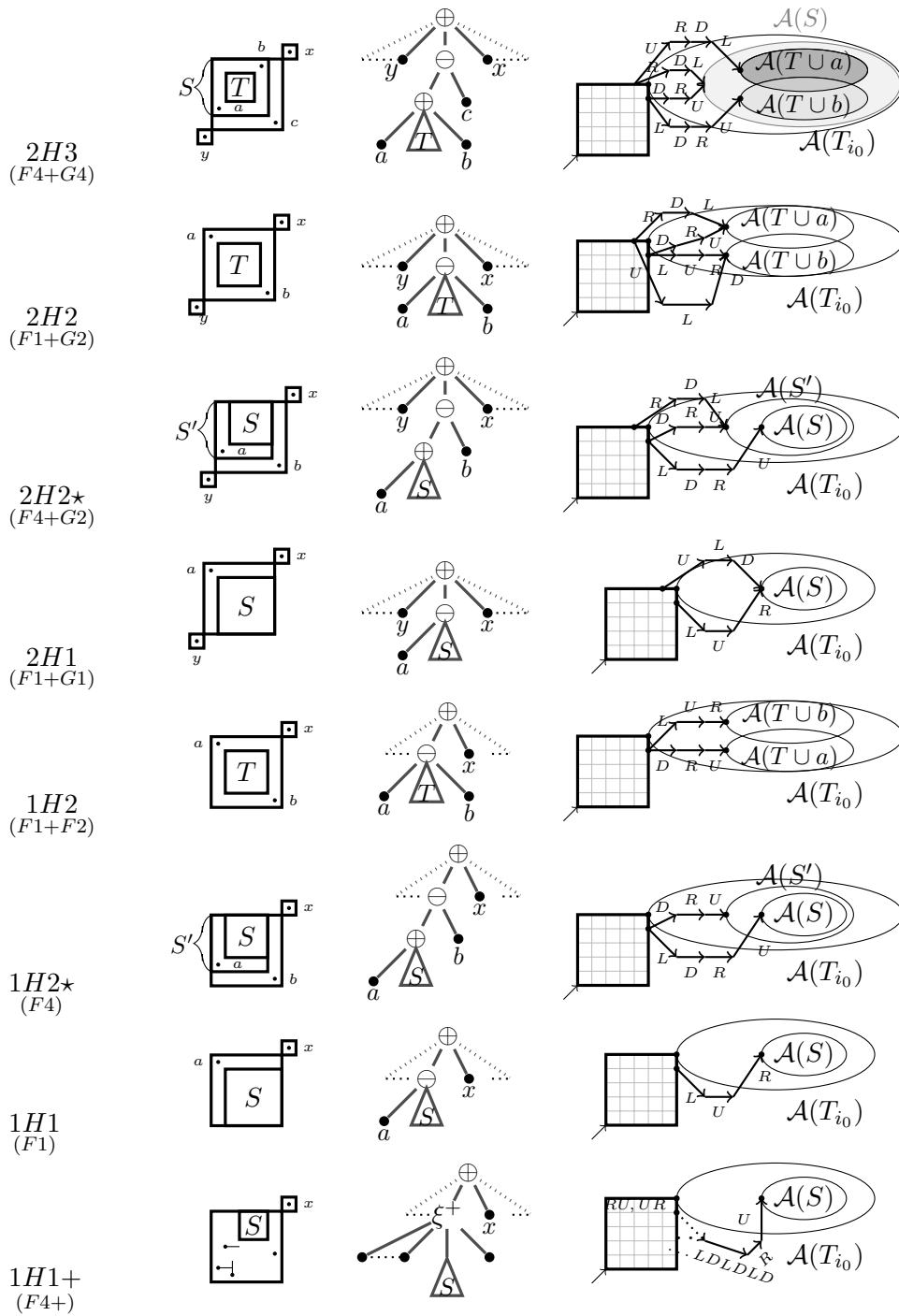
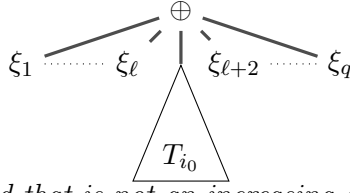


Figure 12: Condition (H) . These subsets form a partition of the set (H) : a permutation satisfies (iH_j) if and only if its diagram has the given shape (maybe with a symmetry) and does not satisfy any condition that appear previously in the list. For example a permutation in $(1H_2)$ can not be in $(2H_2)$.

Theorem 4.17. Let $\sigma =$



be a \oplus -decomposable permutation

where T_{i_0} is the only child that is not an increasing oscillation. Set

$$\mathfrak{P}_{(j)}^{(1)} = (P^{(1)}(\xi_j), P^{(1)}(\xi_{j-1}), \dots, P^{(1)}(\xi_1)) \text{ and } \mathfrak{P}_{(j)}^{(3)} = (P^{(3)}(\xi_j), P^{(3)}(\xi_{j+1}), \dots, P^{(3)}(\xi_q))$$

- If σ does not satisfy any condition (H) then $P(\sigma) = P_0 = P(T_{i_0}) \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$.
- If σ satisfies Condition (1H1) then $P(\sigma) = P_0 \cup P_1$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}.$$

- If σ satisfies Condition (1H1+) then $P(\sigma) = P_0 \cup P_1$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x}_{x \cup T_{i_0}} \cdot w \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)},$$

where w is the unique word encoding the unique reading of the remaining leaves of T_{i_0} .

- If σ satisfies Condition (1H2) then $P(\sigma) = P_0 \cup P_1 \cup P_2$, with

$$P_1 = P(T \cup a) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)} \text{ and } P_2 = P(T \cup b) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}.$$

- If σ satisfies Condition (1H2*) then $P(\sigma) = P_0 \cup P_1 \cup P_2$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)} \text{ and } P_2 = P(S') \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}.$$

- If σ satisfies Condition (2H1) then $P(\sigma) = P_0 \cup P_1 \cup P_2$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)} \text{ and } P_2 = P(S) \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{U}_a}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}.$$

- If σ satisfies Condition (2H2*) then $P(\sigma) = P_0 \cup P_1 \cup P_2 \cup P_3$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}, P_2 = P(S') \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}$$

$$\text{and } P_3 = P(S') \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{R}_b}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}.$$

- If σ satisfies Condition (2H2) then $P(\sigma) = P_0 \cup P_1 \cup P_2 \cup P_3 \cup P_4$, with

$$P_1 = P(T \cup a) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_b}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}, \quad P_2 = P(T \cup b) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)},$$

$$P_3 = P(T \cup a) \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{R}_b}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}, \quad P_4 = P(T \cup b) \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{U}_a}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}.$$

- If σ satisfies Condition (2H3) then $P(\sigma) = P_0 \cup P_1 \cup P_2 \cup P_3 \cup P_4$, with

$$P_1 = P(S) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_c}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}, \quad P_2 = P(T \cup b) \cdot \underbrace{\underbrace{1}_x \cdot \underbrace{D}_c \cdot \underbrace{L}_a}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)},$$

$$P_3 = P(T \cup a) \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{R}_c \cdot \underbrace{U}_b}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}, \quad P_4 = P(S) \cdot \underbrace{\underbrace{3}_y \cdot \underbrace{R}_c}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}.$$

If σ is a \ominus -decomposable permutation, the same holds, up to a change of \oplus into \ominus , increasing into decreasing, $P^{(1)}$ into $P^{(4)}$, $P^{(3)}$ into $P^{(2)}$ and ξ^+ into a decreasing oscillation ξ^- , and a move of the leaf x on the other side of ξ^- .

Proof. It is easy to check that the given pin words are pin words encoding σ . Conversely, we prove that a pin word corresponding to σ is necessarily in $P(\sigma)$. First of all, by [8, Lemma 3.3], every pin representation of the given permutations starts in the only child that is not an oscillation T_{i_0} .

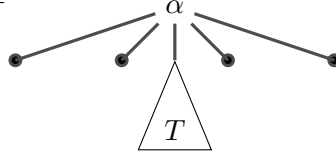
Let us start with the first point of Theorem 4.17. In this case, by Lemma 4.6, we know that T_{i_0} is read in one time. By Lemma 4.4, the other blocks are also read in one time, and Lemma 4.2 ensures that the block closest to T_{i_0} are read first. As there is no relative order between blocks $T_{\ell+2}$ to T_q and blocks T_ℓ to T_1 , this leads to the shuffling operation between pin words corresponding to these blocks, with an external origin placed in quadrant 3 (resp. 1).

In the other cases of Theorem 4.17, by Lemma 4.6, every pin representation of σ either reads T_{i_0} in one time or in two times. In case T_{i_0} is read in one time, the pin representation is as before encoded by pin words of P_0 . If it is read in two times, Lemma 4.6 and its proof and Remark 4.7 ensure that the corresponding pin words are those described.

As example for condition (2H3), P_1 corresponds to condition F2 with $S = T \cup a \cup b$, P_2 corresponds to condition F4 with $S = T \cup b$, P_3 corresponds to condition G3 with $S = T \cup a$ and P_4 corresponds to condition G1 with $S = T \cup a \cup b$. \square

4.4 Recursive case: decomposition trees with a prime root

In the previous section we gave an exact characterization of the pin words associated to a pin-permutation σ whose decomposition tree has a linear root. In this part, we study the recursive case when the decomposition tree has a root which is a simple permutation. First we consider the case where $\sigma =$

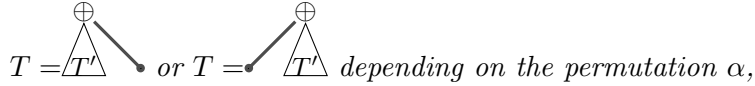


for a non trivial T .

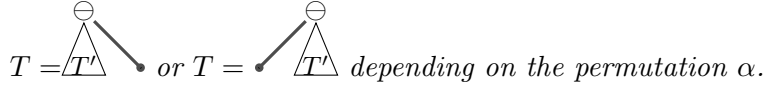
Theorem 4.18. *Let $p = (p_1, \dots, p_n)$ be a pin representation associated to σ . Then p satisfies one of the followings:*

- (1) T is read in one time by p , then $p_1 \in T$.
- (2) T is read in two times by p . Then the second (and last) part of the reading of T consists only of p_n , α is a quasi-oscillation, and the point of α expanded by T is uniquely determined. Moreover,

– if α is an increasing quasi-oscillation, then



– otherwise, α is a decreasing quasi-oscillation, and



Moreover if $p_1 \notin T$ then $T = \{p_2, p_n\}$.

Remark 4.19. *The point of the quasi-oscillation that plays a central role is called the auxiliary point in [8].*

Proof. If $p_1 \notin T$, then by Lemmas 3.12(ii) and 3.9 of [8], $T = \{p_2, p_n\}$. Without loss of generality, assume that $p_1 p_2$ is an increasing subsequence. As $\{p_2, p_n\}$ forms a block, p_n is in one of the 4 positions as shown in Figure 13. But position ③ is forbidden because it is inside of the bounding box \mathcal{B} of $\{p_1, p_2\}$. Positions ② and ④ lie on the side of the bounding box \mathcal{B} . Thus, this point must be read immediately after p_1 and p_2 and thus must be p_3 . But $n > 3$ (α is simple so $|\alpha| \geq 4$) so that this position is forbidden. Hence $T = 12$.

As α is a simple pin-permutation, p_3 respects the separability condition. But if p_3 lies above or on the right of the bounding box then p_n will be on the side of the bounding box of $\{p_1, p_2, p_3\}$, hence $p_n = p_4$. But in that case, α has only 3 children, hence is not simple.

By symmetry we can suppose that p_3 lies on the left of \mathcal{B} . The same argument goes for every pin p_i with $i = 3, \dots, n - 2$ and these pins form an alternating sequence of left and down pins. As p_n separates p_{n-1} from all other pins, p_{n-1} must be an up or right

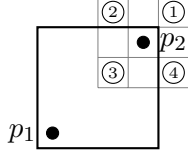


Figure 13: Possible positions for p_n .

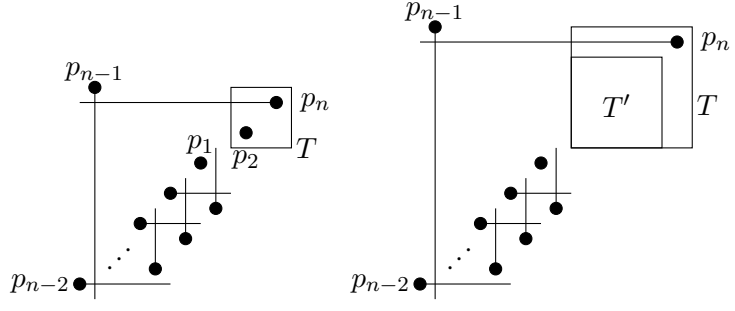


Figure 14: Graphical representation of σ when $p_1 \notin T$ or $p_1 \in T$.

pin (depending on the parity of n). Then α is a quasi-oscillation in which the point expanded by T is uniquely determined and $T = 12$ or $T = 21$ depending on the nature of α -increasing or decreasing-

Suppose now that $p_1 \in T$ but T is not read in one time. By Lemma 3.11(i) of [8], it is read in two times, the second part being p_n . Then $T = \{p_1, \dots, p_i\} \cup \{p_n\}$ with $n \neq i + 1$. Thus p_n does not lie on the side of the bounding box \mathcal{B}' of $\{p_1, \dots, p_i\}$. By symmetry we suppose that p_n lies in quadrant 1 with respect to \mathcal{B}' . As T is a block, no pin must lie on the side of the bounding box of $\{p_1, \dots, p_i, p_n\}$. Thus p_{i+1} does not lie in quadrant 1 with respect to T . If p_{i+1} lies in quadrant 2 or 4, p_n would lie on the side of the bounding box of $\{p_1, \dots, p_{i+1}\}$ and thus must be p_{i+2} . This is in contradiction with the size of α . Thus p_{i+1} lies in quadrant 3 with respect to T , the same goes for p_j with $j \in \{i + 1, \dots, n - 2\}$. Moreover all these pins these pins form an alternating sequence of left and down pins until p_{n-1} which must be an up or right pin depending on the parity of n . Thus α is a quasi-oscillation in which the point expanded by T is uniquely determined and T is \oplus -decomposable if α is increasing. \square

Definition 4.20. For every simple pin-permutation α , with an active point x marked, we define $Q_x(\alpha)$ the set of strict pin words obtained from the set of quasi-strict pin words of α whose first point read in α is x by deleting the first letter of the pin word.

The main result is given in the following theorem:

Theorem 4.21. Let σ be a pin-permutation, whose decomposition tree has a prime root α , with exactly one non trivial child T .

We define Condition (C) as follows

$$(C) \left\{ \begin{array}{l} \alpha \text{ is an increasing (resp. decreasing) quasi-oscillation} \\ T \text{ expands the auxiliary point of } \alpha \\ \text{the shape of } T \text{ is } \begin{array}{c} \oplus \\ \swarrow \quad \searrow \\ \triangleleft \quad \triangleright \end{array} \text{ or } \begin{array}{c} \oplus \\ \swarrow \quad \searrow \\ \triangleright \quad \triangleleft \end{array} \text{ - resp. } \begin{array}{c} \ominus \\ \swarrow \quad \searrow \\ \triangleleft \quad \triangleright \end{array} \text{ or } \begin{array}{c} \ominus \\ \swarrow \quad \searrow \\ \triangleright \quad \triangleleft \end{array} \text{ - where the position} \\ \text{of the leaf is determined by the one of the auxiliary point.} \end{array} \right.$$

Then, denoting by x the point of α expanded by T , the following holds:

- If (\mathcal{C}) is not satisfied, then $P(\sigma) = P(T) \cdot Q_x(\alpha)$.
- If (\mathcal{C}) is satisfied, we distinguish two subcases according to the number of leaves $|T|$ of T :

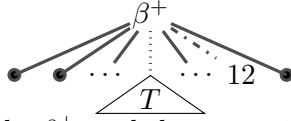
(a) if $|T| \geq 3$, we set $\begin{cases} P_1 = P(T) \cdot Q_x(\alpha) \\ P_2 = P(T') \cdot w \end{cases}$ where w is the unique word encoding the unique reading of the remaining leaves in σ after T' is read when T is read in two times, and then $P(\sigma) = P_1 \cup P_2$.

(b) if $|T| = 2$, we set $P_1 = P(T) \cdot Q_x(\alpha)$, and $P_{\{1,n\}}(\sigma)$ (resp. $P_{\{2,n\}}(\sigma)$) the set of pin words encoding the unique pin representation p of σ such that $T = \{p_1, p_n\}$ (resp. $T = \{p_2, p_n\}$), and then $P(\sigma) = P_1 \cup P_{\{1,n\}} \cup P_{\{2,n\}}$.

Notice that if (\mathcal{C}) is satisfied, then $Q_x(\alpha)$ contains only one word.

It remains to deal with the case where more than one child is non-trivial. From Theorem 3.1 of [8], the root α is an increasing (resp. decreasing) quasi-oscillation, and the node has exactly two children that are not one-point permutations: one of them is the permutation 12 (resp. 21) and both of them expand points of α that are uniquely determined.

Theorem 4.22. Let $\sigma =$



then $P(\sigma) = P(T) \cdot w$ where w is a

word uniquely determined by β^+ and the two points expanded in β^+ and contains only directions.

Proof. Let $p = (p_1, \dots, p_n)$ be a pin representation associated to σ . According to Lemmas 3.11 and 3.12 of [8], p starts with reading T entirely, $T' = 12$ is the second child to be read by p , and p_n is the second point of T' . So there exists k such that $T = \{p_1, \dots, p_k\}$ and $T' = \{p_{k+1}, p_n\}$. Let $q = (p_k, \dots, p_{n-1})$, it is a pin representation of β^+ starting with a fixed active knight. Such a pin representation is unique by Remark 4.8 of [8]. So the points $(p_{k+2}, \dots, p_{n-1})$ are uniquely determined in σ , and p_{k+1} and p_n also (by the position of T). Thus the sequence (p_{k+1}, \dots, p_n) is uniquely determined in σ , and $k \geq 2$ since T is not trivial, and the suffix encoding (p_{k+1}, \dots, p_n) in a pin word of p is a word w uniquely determined by β^+ and the active knight made from the expanded points of β^+ . Since p_{k+1}, \dots, p_n are separating pins, w contains only directions. \square

5 Building automata

In this section, we present a polynomial construction of deterministic automata recognizing the language $\overleftarrow{\mathcal{M}}_\pi$ of words encoding proper pin-permutations σ that contain π as

a pattern. For algorithmic reasons the words w we consider are not the pin words associated to σ , but rather the words $\overleftarrow{\phi(w)}$ where the mirror image \overleftarrow{v} of the word $v = v_1 \dots v_p$ is $v_p \dots v_1$ and ϕ is the mapping given in Definition 3.3 (p.11).

Applying ϕ on pin words transforms the pattern relation on permutations into a piecewise factor relation thanks to Theorem 3.8 (p.12). Moreover we consider mirror images of words in order to preserve determinism. Indeed intuitively the possible beginnings of pin words encoding a permutation may be numerous, whereas the ends of these words are very constrained as it appears in Theorem 4.17 (p.22).

It is important that the construction provides deterministic automata. Given a set B of patterns, this allows us to build a deterministic automaton recognizing words corresponding to proper pin-permutations that contain some $\pi \in B$ as a pattern, the union of languages being obtained by the product of the corresponding automata. Furthermore this deterministic automaton can be complemented in polynomial time, while the same operation on non-deterministic automata is exponential in the worst case. This part will be detailed in Section 6.

By Theorem 3.8 the language $\overleftarrow{\mathcal{M}_\pi}$ can be expressed as $\cup_{u \in P(\pi)} \overleftarrow{\mathcal{L}_u}$. Equivalently $\overleftarrow{\mathcal{M}_\pi} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$ where

$$\overleftarrow{\mathcal{L}_\pi} = \left\{ \overleftarrow{A^* \phi(u^{(1)}) A^* \phi(u^{(2)}) A^* \dots A^* \phi(u^{(j)}) A^*} \mid u \in P(\pi) \text{ and } u = u^{(1)} u^{(2)} \dots u^{(j)} \text{ is the strong numeral led factor decomposition of } u \right\}.$$

The introduction of the language $\overleftarrow{\mathcal{L}_\pi}$ allows us to compute only once the intersection with the language \mathcal{M} instead of once for each element of $P(\pi)$.

In this section, we give an explicit construction of a deterministic automaton \mathcal{A}_π that recognizes the language $\overleftarrow{\mathcal{L}_\pi}$.

5.1 Generic constructions of automata

We define some generic constructions that will be used in the next sections.

Aho-Corasick algorithm. Let X be a finite set of words. Aho-Corasick algorithm [1] builds in linear time and space an automaton that recognizes A^*X . The first step of the algorithm consists in constructing a tree-automaton whose states are labelled by the prefixes of the words of X . The initial state is the empty word ε and the set of final states is made of the words of X . For any word u and any letter a there is a transition labelled by a from state u to state ua if ua is a prefix of a word of X . The second step consists in adding transitions in the automaton to obtain a complete automaton. For any state u and any letter a , the transition from u labelled by a goes to the state corresponding to longest suffix of ua that is also a prefix of a word of X . The set of final state is the set of states corresponding to words ending with a word of X .

A variant for a finite set X . An adaptation of Aho Corasick algorithm allows us to build in linear time the automaton $\mathcal{AC}(X)$ recognizing the set of words ending with a

first occurrence of a word of X . In this construction there is neither outgoing transition from nor loop on the final states. In that case we merge them into a unique final state f . If additionally we add a loop labeled by A on f the language recognized is the set of words having a factor in X .

A variant for a partition X_1, X_2 . When the set X is partitioned into two subsets X_1 and X_2 , we adapt the previous construction and build an automaton $\mathcal{AC}(X_1, X_2)$ with two final states f_1 and f_2 corresponding to the first occurrence of a word of X_1 (resp. X_2).

Concatenation. The concatenation $\mathcal{L}_1 \cdot \mathcal{L}_2$ of two languages respectively recognized by the deterministic automata \mathcal{A}_1 and \mathcal{A}_2 is easy when there is no outgoing transition from the final states of \mathcal{A}_1 . Indeed it is enough to merge the final states of \mathcal{A}_1 with the initial state of \mathcal{A}_2 into a unique state that is neither initial nor final, except when the initial state of \mathcal{A}_2 is final.

Union. Let \mathcal{A}_1 and \mathcal{A}_2 be two deterministic automata such that for any state q there is a path from the initial state to q and a path from q to a final state. We define the automaton $\mathcal{C}(\mathcal{A}_1, \mathcal{A}_2)$ as follows. We perform the cartesian product of \mathcal{A}_1 and \mathcal{A}_2 beginning from the pair of initial states but we stop exploring a path when it enters in a final state of \mathcal{A}_1 or \mathcal{A}_2 . Therefore in $\mathcal{C}(\mathcal{A}_1, \mathcal{A}_2)$ there is no outgoing transitions from any state (q_1, q_2) such that q_1 or q_2 is final. Moreover these states are merged into a unique final state of $\mathcal{C}(\mathcal{A}_1, \mathcal{A}_2)$. Let \mathcal{L}_1 (resp. $\mathcal{L}_2, \mathcal{L}$) be the languages recognized by \mathcal{A}_1 (resp. $\mathcal{A}_2, \mathcal{C}(\mathcal{A}_1, \mathcal{A}_2)$). Then $(\mathcal{L}_1 \cup \mathcal{L}_2)A^* = \mathcal{L}A^*$ and this language is recognized by the automaton $\mathcal{C}(\mathcal{A}_1, \mathcal{A}_2)$ with an additional loop labeled by A on the final state. Notice that the automaton obtained is deterministic and that the complexity in time and space of this construction is proportional to the product of the number of states of the automata \mathcal{A}_1 and \mathcal{A}_2 .

5.2 Non-recursive cases

Pin-permutation of size 1. For the permutation $\pi = 1$, we have $P(\pi) = \{1, 2, 3, 4\}$. Then $\overleftarrow{\mathcal{L}}_\pi = \{\overleftarrow{A^* \phi(w) A^*} \mid w \in P(\pi)\} = A^* \{UR, UL, DR, DL, RU, RD, LU, LD\} A^*$. The language $\overleftarrow{\mathcal{L}}_\pi$ is the one recognized by the automaton of Figure 15.

Simple pin-permutations. Let π be a simple permutation of size n . By Remarks 2.1 and 2.2 (p.4 and p.5) the pin words encoding π are either strict or quasi-strict. Therefore

$$\overleftarrow{\mathcal{L}}_\pi = \bigcup_{\substack{u \in P(\pi) \\ u \text{ strict}}} \overleftarrow{A^* \phi(u) A^*} \quad \bigcup_{\substack{u = u^{(1)} u^{(2)} \in P(\pi) \\ u \text{ quasi-strict}}} \overleftarrow{A^* \phi(u^{(1)}) A^* \phi(u^{(2)}) A^*}.$$

Note that the cardinality of $P(\pi)$ is smaller than 48×8 for any simple permutation π (see [7, Lemma 4.2]). Moreover for any quasi-strict pin word $u^{(1)}u^{(2)}$ of $P(\pi)$, we have

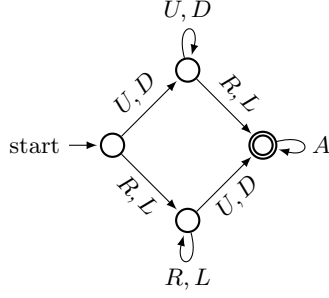


Figure 15: The automaton \mathcal{A}_π when $\pi = 1$.

$iu^{(2)} \in P(\pi)$ for all $i \in \{1, \dots, 4\}$. This allows to describe the language $\overleftarrow{\mathcal{L}}_\pi$ as :

$$\overleftarrow{\mathcal{L}}_\pi = \left(\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict}}} A^* \overleftarrow{\phi(u)} \quad \bigcup \left(\bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict}}} A^* \overleftarrow{\phi(u^{(2)})} \cdot \bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict}}} A^* \overleftarrow{\phi(u^{(1)})} \right) \right) A^*.$$

Let $E_1 = \{\overleftarrow{\phi(u)} \mid u \in P(\pi), u \text{ is strict}\}$, $E_2 = \{\overleftarrow{\phi(u^{(2)})} \mid u = u^{(1)}u^{(2)} \in P(\pi), u \text{ is quasi-strict}\}$ and $E_3 = \{\overleftarrow{\phi(u^{(1)})} \mid u = u^{(1)}u^{(2)} \in P(\pi), u \text{ is quasi-strict}\} = A^2 \cap \mathcal{M}$. Denote by $\mathcal{AC}(E_1)$, $\mathcal{AC}(E_2)$ and $\mathcal{AC}(E_3)$ the automata recognizing respectively the set of words ending with a first occurrence of a word of E_1 , E_2 and E_3 and obtained following the construction given in Section 5.1. Note that the size of the two first automata is linear in the size n of π and that the one of the last one is constant.

The deterministic automaton \mathcal{A}_π is finally obtained by concatenation of the automata of $\mathcal{AC}(E_2)$ and $\mathcal{AC}(E_3)$ whose size is still linear followed by the union with the automaton $\mathcal{AC}(E_1)$. Finally a loop labeled with A is added on the final state. Hence the size of the automaton \mathcal{A}_π is quadratic in n .

Remark 5.1. *This construction is based on the partition of $P(\pi)$ into strict and quasi-strict pin words. It can be used for any permutation π whose pin words are either strict or quasi-strict even if π is not simple.*

Notice that for the special case of simple permutations we gave in [7] a construction of an automaton recognizing a language \mathcal{L} such that $\mathcal{L} \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$. The size of this automaton is linear in the size of π , but it relies on an *ad hoc* description of $\overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$ when π is simple.

Pin-permutations with a linear root. Without loss of generality the only non-recursive case is the one where $\pi = \oplus[\xi_1, \dots, \xi_q]$, every ξ_j being an oscillation. Theorem 4.16 (p.20) gives an explicit description of the elements of $P(\pi)$. These words are the concatenation of a strict or a quasi-strict pin word belonging to some $P^{double}(\xi_j, \xi_{j+1})$

with a sequence of pin words belonging to the shuffle product

$$(P^{(1)}(\xi_{j-1}), \dots, P^{(1)}(\xi_1)) \sqcup\sqcup (P^{(3)}(\xi_{j+2}), \dots, P^{(3)}(\xi_q)).$$

Notice that from Lemma 4.10 (p.18) for all i the pin words of respectively $P^{(1)}(\xi_i)$ and $P^{(3)}(\xi_i)$ are strict. This will ease the decomposition of $u \in P(\pi)$ into strong numeral-led factors that is needed to describe $\overleftarrow{\mathcal{L}}_\pi$.

With the construction given in Section 5.1 we build deterministic automata

$$\mathcal{A}(\xi_i, \xi_j) = \mathcal{AC}(\overleftarrow{\phi(P^{(1)}(\xi_i))}, \overleftarrow{\phi(P^{(3)}(\xi_j))}).$$

This construction is made possible since, for all i, j , $\phi(P^{(1)}(\xi_i)) \cap \phi(P^{(3)}(\xi_j))$ is empty, these two languages being defined on disjoint alphabets. We denote by s_{ij} (resp. $f_{ij}^{(1)}$, $f_{ij}^{(3)}$) the initial state (resp. final state corresponding to the first occurrence of a word of $\overleftarrow{\phi(P^{(1)}(\xi_i))}$, resp. of $\overleftarrow{\phi(P^{(3)}(\xi_j))}$) of the automaton $\mathcal{A}(\xi_i, \xi_j)$.

Lemma 4.15 (p.20) ensures that for any j , the pin words in $P^{double}(\xi_j, \xi_{j+1}) = P(\oplus[\xi_j, \xi_{j+1}])$ are either strict or quasi-strict. As noticed in Remark 5.1, this allows us to build a deterministic automaton $\mathcal{A}^{double}(\xi_j, \xi_{j+1})$ recognizing $\overleftarrow{\mathcal{L}}_{\oplus[\xi_j, \xi_{j+1}]}$ as in the case of $\overleftarrow{\mathcal{L}}_\pi$ for simple permutations π . The unique final state f_j of this automaton has no outgoing transitions except for the loop labeled by the alphabet A .

With the description of $P(\pi)$ as a shuffle product in Theorem 4.16, these automata can be glued together to finish the construction of \mathcal{A}_π , as shown in Figure 17. To avoid a blow-up in the construction, we proceed as follows. For any i, j such that $1 \leq i < j \leq q$

- if $i + 1 \neq j$, then s_{ij} , $f_{(i-1)j}^{(1)}$ and $f_{i(j+1)}^{(3)}$ are merged into a unique state that is neither initial (except when $i = 1$ and $j = q$) nor final,
- if $i + 1 = j$, $f_{(i-1)j}^{(1)}$, $f_{i(j+1)}^{(3)}$ and the initial state of $\mathcal{A}^{double}(\xi_i, \xi_j)$ are merged into a unique state that is neither initial nor final,

and the final states f_j are merged into a unique final state having a loop labeled by A . The first item above corresponds to the shuffle product construction and the second one to the concatenation with a word recognized by some \mathcal{A}^{double} . To be more precise, the language accepted by the merged state $s_{i,j}$ is

$$\bigcup_{\substack{(w_1, w_2, \dots, w_{q-j+i-1}) \in \\ (P^{(1)}(\xi_{i-1}), \dots, P^{(1)}(\xi_1)) \sqcup\sqcup (P^{(3)}(\xi_{j+1}), \dots, P^{(3)}(\xi_q))}} \overleftarrow{\phi(w_1)A^*\phi(w_2)A^* \dots A^*\phi(w_{q-j+i-1})A^*}.$$

The size of each automaton $\mathcal{A}(\xi_i, \xi_j)$ is linear in the size n of π , and $\mathcal{O}(n^2)$ such automata are involved in the construction of \mathcal{A}_π . The automata $\mathcal{A}^{double}(\xi_j, \xi_{j+1})$ are each of quadratic size, as in the case of simple pin-permutations, but there are only $q - 1 = \mathcal{O}(n)$ of them in \mathcal{A}_π . Therefore, the size of \mathcal{A}_π is cubic in n .

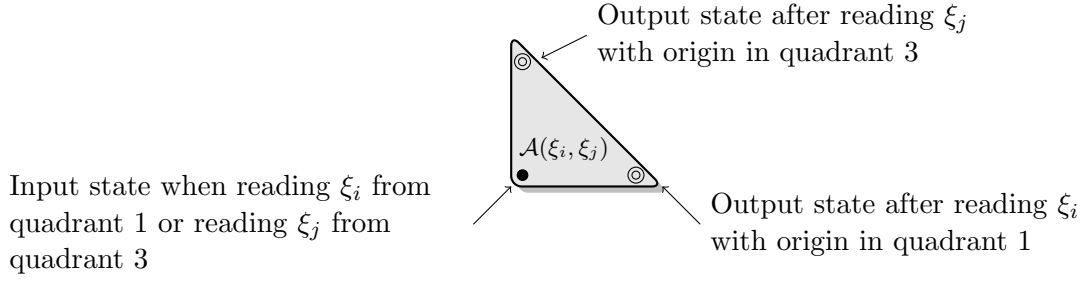


Figure 16: Atomic automaton $\mathcal{A}(\xi_i, \xi_j)$ used in the construction of \mathcal{A}_π .

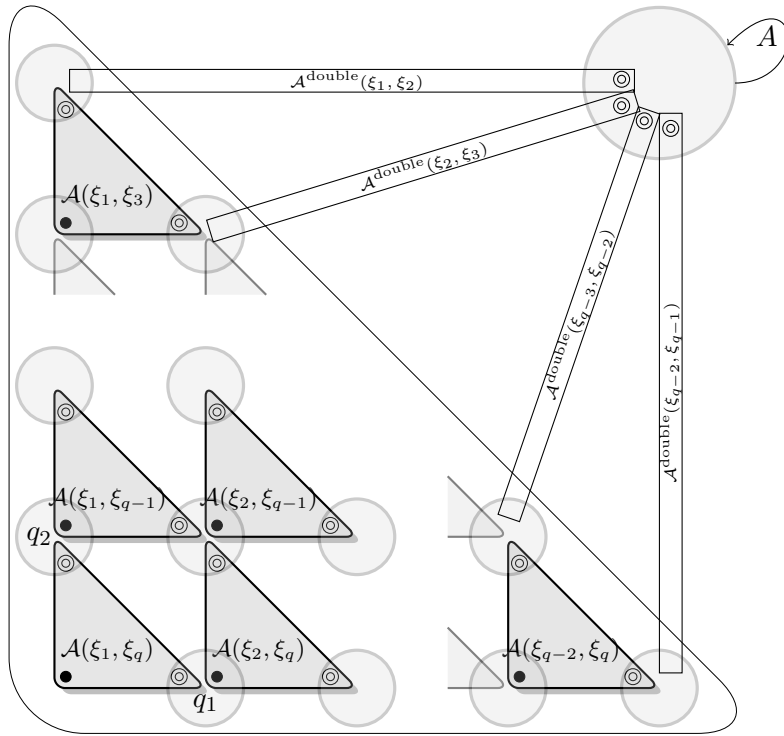
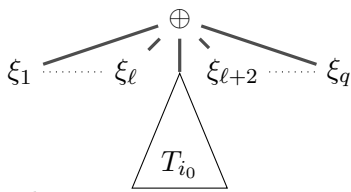


Figure 17: Automaton \mathcal{A}_π for $\pi = \oplus[\xi_1, \dots, \xi_q]$ where every ξ_i is an increasing oscillation.

5.3 Recursive decomposition tree with linear root

Suppose without loss of generality that the root has label \oplus . Every child of the root is an increasing oscillation, except exactly one, denoted T_{i_0} . The decomposition tree of

π is  . The automaton $\mathcal{A}(T_{i_0}) = \mathcal{A}_\tau$ of the permutation τ whose decomposition tree is T_{i_0} can be recursively obtained.

If π does not satisfy any condition (H) of Figure 12 (p.21), then Theorem 4.17 (p.22) ensures that $P(\pi) = P(\tau) \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$ with

$$\mathfrak{P}_{(\ell)}^{(1)} = (P^{(1)}(\xi_\ell), P^{(1)}(\xi_{\ell-1}), \dots, P^{(1)}(\xi_1)) \text{ and } \mathfrak{P}_{(\ell+2)}^{(3)} = (P^{(3)}(\xi_{\ell+2}), P^{(3)}(\xi_{\ell+3}), \dots, P^{(3)}(\xi_q)).$$

To deal with the shuffle product, we use again the automata $\mathcal{A}(\xi_i, \xi_j)$ whose initial and final states are denoted $s_{ij}, f_{ij}^{(1)}$ and $f_{ij}^{(3)}$ as before. We furthermore introduce deterministic automata $\mathcal{A}^{(1)}(\xi_j) = \mathcal{AC}(\overleftarrow{\phi(P^{(1)}(\xi_j))})$ and $\mathcal{A}^{(3)}(\xi_j) = \mathcal{AC}(\overleftarrow{\phi(P^{(3)}(\xi_j))})$ whose initial and final states are denoted respectively $s_j^{(1)}, f_j^{(1)}, s_j^{(3)}$ and $f_j^{(3)}$. The automata $\mathcal{A}^{(1)}(\xi_j)$ (resp. $\mathcal{A}^{(3)}(\xi_j)$) correspond to the reading of parts of $\mathfrak{P}_{(\ell)}^{(1)}$ (resp. $\mathfrak{P}_{(\ell+2)}^{(3)}$) when the reading of $\mathfrak{P}_{(\ell+2)}^{(3)}$ (resp. $\mathfrak{P}_{(\ell)}^{(1)}$) is completed.

The language \mathcal{L}_π associated to π is the language of pin words recognized by the automaton given in Figure 18 where the following merges are performed:

- for any i, j such that $1 \leq i \leq \ell$ and $\ell+2 \leq j \leq q$, $s_{ij}, f_{(i-1)j}^{(1)}$ and $f_{i(j+1)}^{(3)}$ are merged into a unique state that is neither initial (except when $i = 1$ and $j = q$) nor final,
- for $1 \leq i \leq \ell$, $s_i^{(1)}, f_{(i-1)}^{(1)}$ and $f_{i(\ell+2)}^{(3)}$ are merged into a unique state that is neither nor final,
- for $\ell+2 \leq j \leq q$, $s_j^{(3)}, f_{j+1}^{(3)}$ and $f_{\ell j}^{(3)}$ are merged into a unique state that is neither nor final,
- $f_{\ell+2}^{(3)}, f_\ell^{(1)}$ and the initial state of \mathcal{A}_τ are merged into a unique state that is neither nor final.

If one of the conditions given in Figure 12 (p.21) holds for π , the automaton has the same general structure but some transitions are added as depicted in Figure 12 (p.21) where the squares are compact representations for shuffle automata given in Figure 18. More precisely these new transitions form paths beginning in hatched states of Figure 18 and arriving in marked states of \mathcal{A}_τ that are initial states of subautomata as shown in Figure 12. The way the states of \mathcal{A}_τ are marked is explicited in Section 5.5. Moreover as the transitions leaving hatched states are labeled by directions that never appear in $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$), the added transitions do not affect the determinism of the automata.

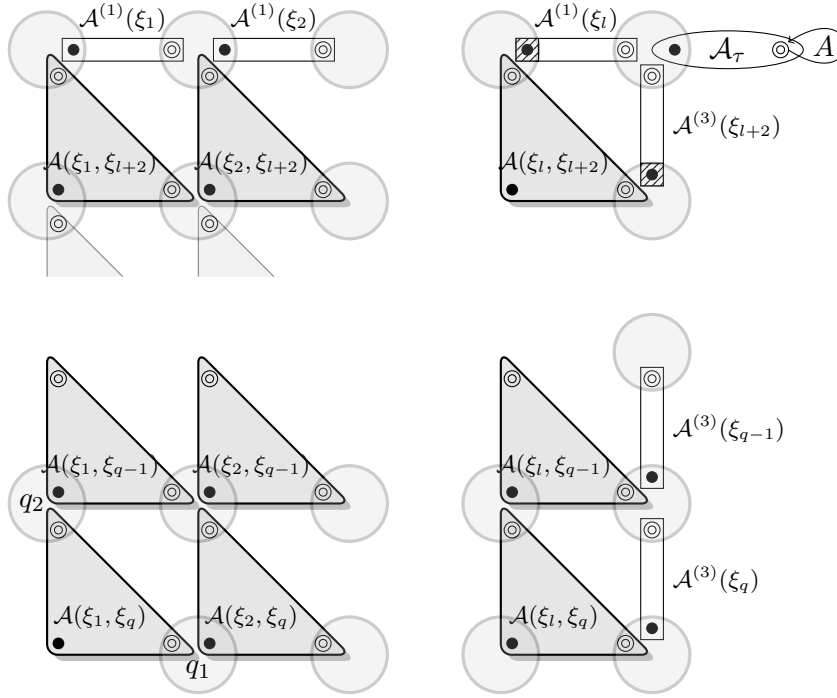
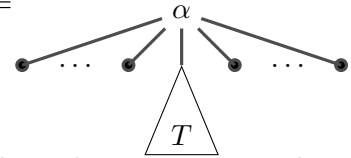


Figure 18: The automaton \mathcal{A}_π for $\pi = \oplus[\xi_1, \dots, \xi_\ell, \tau, \xi_{\ell+2}, \dots, \xi_q]$, where every ξ_i is an increasing oscillation.

The additional (after the recursive part) time and space complexity for building this automaton is $\mathcal{O}(n^2)$.

5.4 Decomposition tree whose root is a simple permutation

Exactly one child of the root is not a leaf. Let $\pi =$



where α is a simple permutation and T is not a leaf. Denote by τ the permutation whose decomposition tree is T .

We recall condition \mathcal{C} introduced in Theorem 4.21 (p.25):

$$(\mathcal{C}) \left\{ \begin{array}{l} \alpha \text{ is an increasing (resp. decreasing) quasi-oscillation} \\ T \text{ expands the auxiliary point of } \alpha \\ \text{the shape of } T \text{ is } \begin{array}{c} \oplus \\ \diagup \quad \diagdown \\ \triangle \\ \diagdown \quad \diagup \end{array} \text{ or } \begin{array}{c} \oplus \\ \diagdown \quad \diagup \\ \triangle \\ \diagup \quad \diagdown \end{array} \text{ - resp. } \begin{array}{c} \ominus \\ \diagup \quad \diagdown \\ \triangle \\ \diagdown \quad \diagup \end{array} \text{ or } \begin{array}{c} \ominus \\ \diagdown \quad \diagup \\ \triangle \\ \diagup \quad \diagdown \end{array} \text{ - where the position} \\ \text{of the leaf is determined by the one of the auxiliary point.} \end{array} \right.$$

Recall also that $Q_x(\alpha)$ denotes the set of strict pin words obtained from the set of quasi-strict pin words of α whose first point read in α is x by deleting the first letter of the pin word. In particular the cardinality of $Q_x(\alpha)$ is smaller than the one of $P(\alpha)$, hence smaller than 48×8 (see [7, Lemma 4.2]). Moreover $Q_x(\alpha)$ can be determined in linear time: indeed it is sufficient to examine the proper pin representations which start with an active knight containing x .

If \mathcal{C} is not satisfied then from Theorem 4.21 $P(\pi) = P(\tau) \cdot Q_x(\alpha)$ and the automaton recognizing $\overleftarrow{\mathcal{L}}_\pi$ is obtained by the concatenation of $\mathcal{AC}(\overleftarrow{\phi}(Q_x(\alpha)))$ with $\mathcal{A}(T) = \mathcal{A}_\tau$.

When \mathcal{C} is satisfied and $|T| \geq 3$, then by Theorem 4.21 $P(\pi)$ contains $P(\tau) \cdot Q_x(\alpha)$ and some other words. These other words are $P(T') \cdot w$ where w is the unique word encoding the unique reading of the remaining leaves in π after T' is read when T is read in two times. As in the case of a linear root, we first build the automaton \mathcal{A} that is the concatenation of the automaton $\mathcal{AC}(\overleftarrow{\phi}(Q_x(\alpha)))$ with \mathcal{A}_τ and then we add some new transitions to it to account for the words in $P(T') \cdot w$. To do so, we read the word $\overleftarrow{\phi}(w)$ in \mathcal{A} as long as states are visited for the first time. The first transition that reaches a state already visited is transformed into a transition to a new state from which new states are created to finish the reading of $\overleftarrow{\phi}(w)$. The path so obtained ends in a marked state (see Section 5.5) of \mathcal{A}_τ that is the initial state of $\mathcal{A}(T')$. Like in Aho-Corasick algorithm any state q of $\mathcal{AC}(\overleftarrow{\phi}(Q_x(\alpha)))$ is labelled by the shortest word labelling a path from the initial state to q . Moreover we complete the construction by adding transitions from the states newly created: for any such state q , the transition from q labelled by a goes to the longest suffix of $q \cdot a$ that is a state of the automaton. As in the Aho-Corasick algorithm, this construction is linear.

When \mathcal{C} is satisfied and $|T| = 2$, the construction is no more recursive. Permutation π is explicit and so are its pin words. Indeed from Theorem 4.21 such a permutation has exactly 4 pin representations, hence at most 32 pin words that are either strict, quasi-strict or start with the reading of T . In the third case, the third letter is a numeral. We distinguish two cases depending on the second letter that is either a numeral or a direction.

The pin words beginning with three numerals can be partitioned into two families \mathcal{F} and \mathcal{F}' corresponding to the two pin representations of π starting with the reading of T . For a word $u = u^{(1)}u^{(2)}u^{(3)}$ in \mathcal{F} (resp. \mathcal{F}'), $u^{(2)}$ and $u^{(3)}$ are uniquely determined and $u^{(1)}$ can be any numeral.

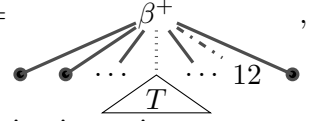
For any pin word $u = u^{(1)}u^{(2)}$ of π , where $u^{(1)}$ is of length 2 and any strict pin word $v^{(1)}$ encoding T , the word $v^{(1)}u^{(2)}$ is also a pin word of π .

Therefore

$$\begin{aligned}
\overleftarrow{\mathcal{L}}_\pi = & \left(\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict}}} A^* \overleftarrow{\phi}(u) \quad \bigcup \left(\bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict and } |u^{(1)}|=1}} A^* \overleftarrow{\phi}(u^{(2)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict and } |u^{(1)}|=1}} A^* \overleftarrow{\phi}(u^{(1)}) \right) \right. \\
& \bigcup \left(\bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict and } |u^{(1)}|=2}} A^* \overleftarrow{\phi}(u^{(2)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict and } |u^{(1)}|=2}} A^* \overleftarrow{\phi}(u^{(1)}) \right) \\
& \bigcup \left(\bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}}} A^* \overleftarrow{\phi}(u^{(3)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}}} A^* \overleftarrow{\phi}(u^{(2)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}}} A^* \overleftarrow{\phi}(u^{(1)}) \right) \\
& \left. \bigcup \left(\bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}'}} A^* \overleftarrow{\phi}(u^{(3)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}'}} A^* \overleftarrow{\phi}(u^{(2)}) \cdot \quad \bigcup_{\substack{u=u^{(1)}u^{(2)}u^{(3)} \in P(\pi) \\ u \in \mathcal{F}'}} A^* \overleftarrow{\phi}(u^{(1)}) \right) \right) A^*.
\end{aligned}$$

As in the non recursive case for simple pin-permutations we compute a deterministic automaton recognizing $\overleftarrow{\mathcal{L}}_\pi$ making use of concatenations, unions and Aho-Corasick like constructions. As there are four terms in the union, the complexity of the construction is $\mathcal{O}(n^4)$ in time and space.

Two children are not singletons. Up to symmetry this means that $\pi =$



where β^+ is an increasing quasi-oscillation, and T expands the main substitution point and 12 the auxiliary one.

Theorem 4.22 ensures that the pin words encoding π are of the form vw where $v \in P(T)$, w uniquely determined by β^+ and containing only directions. Therefore $\phi(w) = w$ and $\overleftarrow{\mathcal{L}}_\pi = A^* \overleftarrow{\phi}(w) \overleftarrow{\mathcal{L}}_\tau$ where τ is the permutation whose decomposition tree is T . The automaton recognizing $\overleftarrow{\mathcal{L}}_\pi$ is the concatenation of $\mathcal{AC}(\overleftarrow{\phi}(w))$ with \mathcal{A}_τ .

This construction can be done in linear time and space in addition to the time and space complexity of the construction of \mathcal{A}_τ .

5.5 Marking initial states

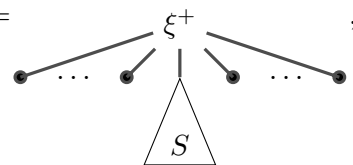
To finish the construction, we must explain how the different states of the automata are marked.

The need of creating a transition to a marked state (of a subautomaton) happens only when building the automaton \mathcal{A}_π for a permutation π whose decomposition tree has a prime root and satisfies Condition (C) with $|T| \geq 3$, or for a permutation π whose decomposition tree has a linear root and satisfies a condition (H) of Figure 12.

As it appears in Figure 12 and Condition (C), in almost all such situations, the marked state belongs to a subautomaton corresponding to a permutation τ whose decomposition

tree has a linear root. There is only one situation where this root is prime: when π satisfies condition $1H1+$.

In this case, we need to mark in the automaton associated to $T =$



the state q such that when starting from q the language recognized is the one recognized by $\mathcal{A}(S)$.

The automaton $\mathcal{A}(T)$ is obtained as described in Section 5.4, when exactly one child of the root is not a leaf (indeed $|S| \geq 2$). The marking of state q depends on how the automaton $\mathcal{A}(T)$ is built and in particular on whether Condition (\mathcal{C}) is satisfied or not.

Recall that ξ^+ is an increasing oscillation. If ξ^+ has a size at least 5, it is not a quasi-oscillation, and Condition \mathcal{C} is not satisfied. Therefore $\mathcal{A}(T)$ is the concatenation of two automata the second of which is $\mathcal{A}(S)$ whose initial state can be readily marked.

If ξ^+ has size 4, then $\xi^+ = 2413$ is a quasi-oscillation and Condition (\mathcal{C}) may be satisfied. If it is not the case, $\mathcal{A}(T)$ is obtained as above and so is the marking of state q . If on the contrary Condition (\mathcal{C}) is satisfied, the construction of $\mathcal{A}(T)$ depends on $|S|$. If $|S| \geq 3$, $\mathcal{A}(T)$ is again the concatenation of two automata but some states and transitions have been added that do not enter the initial state of S that we mark as above. If $|S| = 2$, then T has size 5 and is equal to 23514 or 25134 . It is straightforward to draw the automaton and mark the state.

Otherwise consider a permutation τ whose decomposition tree T has a linear root. The need of a marked state in \mathcal{A}_τ corresponds to the case where the leftmost (resp. rightmost) child of T is a leaf ℓ . Each marked state q is such that the language accepted starting from q is the set of words encoding the readings of all nodes of T except the leaf ℓ . There are at most two such leaves and the corresponding marked states are indicated as q_1 and q_2 in Figures 17 and 18.

Notice that in Conditions $(2H3)$, $(2H2\star)$ and $(1H2\star)$ two recursive levels of marked states have to be stored. In the overall construction we never use more than two levels of marked states. Therefore the markings coming from deeper levels of recursion become obsolete and can be forgotten.

5.6 Complexity analysis

Proposition 5.2. *For every permutation π of size n , the preceding procedure running in time $\mathcal{O}(n^4)$ outputs a deterministic automaton recognizing the language of $\overleftarrow{\mathcal{L}}_\pi$. The size of this automaton is $\mathcal{O}(n^4)$.*

Proposition 5.2 easily follows from the complexities of the previous constructions that are summarized in the following array:

permutation	Time complexity	Size of the automaton
size 1	$\mathcal{O}(1)$	$\mathcal{O}(1)$
simple	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
root \oplus non-recursive	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
root \oplus recursive, one child T is not an oscillation	$\mathcal{O}(n^2)$ + time for the construction on T	$\mathcal{O}(n^2)$ + size of the automaton for T
root is prime recursive, \mathcal{C} satisfied, and T has size 2	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$
root is prime recursive, one child T is not a leaf (if not preceding case)	$\mathcal{O}(n)$ + time for the construction on T	$\mathcal{O}(n)$ + size the automaton for T
root prime recursive, and two children not leaves	$\mathcal{O}(n)$ + time for the construction on T	$\mathcal{O}(n)$ + size of the automaton for T

Notice that no extra time is needed to mark the states of the automaton.

6 A polynomial algorithm for permutation classes

In this section, we detail the whole algorithm that determines whether a permutation class $\mathcal{C} = Av(B)$, where B is finite, contains a finite number of simple permutations. Denoting $k = |B|$ and $n = \sum_{\pi \in B} |\pi|$, the complexity of the algorithm is polynomial w.r.t. n and exponential w.r.t. k . The algorithm can be decomposed into several steps.

Parallel alternations and wedge simple permutations. Following [13] (see Theorem 2.4 p. 6) we first check if \mathcal{C} contains arbitrarily long parallel alternations or wedge simple permutations of type 1 or 2. From Lemmas 2.5, 2.6, 2.7 this problem is equivalent to testing if permutations of B contain some patterns of size at most 4. Using a result of [2], this can be done in $\mathcal{O}(n \log n)$ time.

Pin-permutation recognition. The next step is to determine the subset $PB \subseteq B$ of pin-permutations of B . To do so we use the characterization of the class of pin-permutations by their decomposition trees established in [8]. More precisely, given $\pi \in B$, we compute its decomposition tree T_π in linear time following [9]. Then for each simple permutation σ labeling a node of T_π we check if σ is a *pin*-permutation by computing with Algorithm 2 of [7] its set of associated pin words and testing its emptiness. Then we determine if the shape of the tree is of the shape described in [8] by a linear time depth first traversal of T_π . As the complexity of each step is linear, deciding whether a permutation is a pin-permutation or not can be done in time linear with respect to the size of the permutation. Therefore the overall determination of PB is linear in n .

Building the final deterministic automaton. For each pin-permutation $\pi \in PB$, we construct \mathcal{A}_π in time at most $|\pi|^4$ following Section 5. Then we build a deterministic automaton \mathcal{A}'' which is the union of all these automata \mathcal{A}_π for $\pi \in PB$ in time

$\mathcal{O}(\max\{|\pi, \pi \in PB|\}^{4|PB|})$. Notice that this union is performed using a product of automata, in order to preserve determinism. Then we build the automaton \mathcal{A}' which is the deterministic intersection between \mathcal{A}'' and the automaton $\mathcal{A}(\mathcal{M})$ given in Figure 19 in time $\mathcal{O}(|\mathcal{A}'| \cdot |\mathcal{A}(\mathcal{M})|) = \mathcal{O}(|\mathcal{A}'|)$.

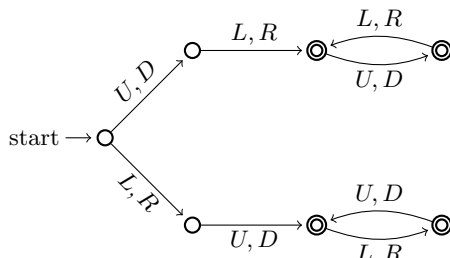


Figure 19: A deterministic automaton $\mathcal{A}(\mathcal{M})$ recognizing the set \mathcal{M} of words of length at least 2 without factor in $\{UU, UD, DU, DD, RR, RL, LR, LL\}$.

Then \mathcal{A}' recognizes $\overleftarrow{\phi(w)}$ for all strict pin words w encoding a permutation (which is necessarily a proper pin-permutation) which is not in \mathcal{C} . Next we complement this automaton to build \mathcal{A}'^c and compute again its intersection with $\mathcal{A}(\mathcal{M})$ to obtain the automaton \mathcal{A}^c . This automaton recognizes $\overleftarrow{\phi(w)}$ for all strict pin words w that encode a permutation of \mathcal{C} (that is again necessarily a proper pin-permutation). These last operations are performed in linear time with respect to the size of $|\mathcal{A}'|$.

Then checking if the permutation class \mathcal{C} contains an infinite number of proper pin-permutations is equivalent to checking if the language recognized by \mathcal{A}^c is infinite i.e. if \mathcal{A}^c contains a cycle that is accessible and co-accessible. Notice that in the construction every state in \mathcal{A}^c is accessible and co-accessible, hence it is enough to test whether \mathcal{A}^c contains a cycle.

Putting all these steps together leads to an algorithm whose complexity is $\mathcal{O}(n^{4k})$ where n is maximal size of a pin-permutation of B and k the number of pin-permutations in B .

7 Conclusion

The work reported here follows the line opened by [3] and continued by [13]. In [3], the main theorem provides (in particular) a sufficient condition for a permutation class \mathcal{C} to have an algebraic generating function: namely, that \mathcal{C} contains a finite number of simple permutations. Then, [13] introduces new objects (most importantly, pin-permutations) to provide a decision procedure testing this sufficient condition, for classes with a finite and explicit basis. Making use of the detailed study of pin-permutations in [8], we described in the above a more efficient algorithm testing this condition.

However, when the number of simple permutations in \mathcal{C} is finite, the above procedures do not allow to *compute* the finite set $\mathcal{S}_{\mathcal{C}}$ of simple permutations in \mathcal{C} . A method to do

so is described in [3], but it is of highly exponential complexity. A first open problem is to find an efficient algorithm to compute $\mathcal{S}_{\mathcal{C}}$. We believe this is both a natural question emerging from our work, and a stepping stone for future research.

Indeed, the proof of the main theorem of [3] is constructive, meaning that it gives a method to compute from the set $\mathcal{S}_{\mathcal{C}}$ an algebraic system of equations satisfied by the generating function of \mathcal{C} . The main step is actually to compute a combinatorial specification (in the sense of [15], here a context-free grammar of trees) for the permutations of \mathcal{C} , or rather their decomposition trees. A second task for future research would then be to transform the proof of the theorem of [3] into an algorithm computing this combinatorial specification for \mathcal{C} from the set $\mathcal{S}_{\mathcal{C}}$ of its simple permutations (that we assume to be finite and explicit, or hopefully computed by an algorithm as described in the previous paragraph).

From such a specification, it is of course possible with the methodology of [15] to immediately deduce a system of equations for the generating function of \mathcal{C} , as done in [3]. But other developments can be considered. In particular, this opens the way to obtaining systematically Boltzmann random samplers of permutations in a class, or to the automatic evaluation of the Stanley-Wilf growth rate of a class. If algorithms answering these questions were designed and implemented, they would be very valuable tools for the community to attack well-known conjectures on the subject.

References

- [1] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6), June 1975.
- [2] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366, London, UK, 2001. Springer-Verlag.
- [3] Michael H. Albert and Mike D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1-3):1–15, 2005.
- [4] Michael H. Albert, Steve Linton, and Nik Ruškuc. The insertion encoding of permutations. *Electron. J. Combin.*, 12:Research Paper 47, 31 pages (electronic), 2005.
- [5] Mike D. Atkinson, Nik Ruškuc, and Rebecca Smith. Substitution-closed pattern classes. *Journal of Combinatorial Theory Series A*, 2011.
- [6] Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, and Dominique Rossin. Deciding the finiteness of simple permutations contained in a wreath-closed class is polynomial. In *Proceedings of Permutation Patterns 2009 (PP 2009)*, pages 13–20, July 2009.

- [7] Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, and Dominique Rossin. Deciding the finiteness of the number of simple permutations contained in a wreath-closed class is polynomial. *Pure Mathematics and Applications*, 21(2):119–135, 2010.
- [8] Frédérique Bassino, Mathilde Bouvel, and Dominique Rossin. Enumeration of pin-permutations. *The Electronic Journal of Combinatorics*, 2011. 39 pages.
- [9] Anne Bergeron, Cédric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. In *European Symposium on Algorithm (ESA 2005)*, pages 779–790, 2005.
- [10] Mireille Bousquet-Mélou. Four classes of pattern-avoiding permutations under one roof: Generating trees with two labels. *Electr. J. Comb.*, on(2), 2002.
- [11] Robert Brignall, Sophie Huczynska, and Vincent Vatter. Decomposing simple permutations, with enumerative consequences. *Combinatorica*, 28(4):385–400, jul 2008.
- [12] Robert Brignall, Sophie Huczynska, and Vincent Vatter. Simple permutations and algebraic generating functions. *J. Combin. Theory Ser. A*, 115(3):423–441, 2008.
- [13] Robert Brignall, Nik Ruškuc, and Vincent Vatter. Simple permutations: decidability and unavoidable substructures. *Theoret. Comput. Sci.*, 391(1-2):150–163, 2008.
- [14] Sergi Elizalde. *Statistics on pattern-avoiding permutations*. PhD thesis, MIT, 2004.
- [15] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.
- [16] Steffen Heber and Jens Stoye. Finding all common intervals of k permutations. In *12th Annual Symposium Combinatorial Pattern Matching, (CPM 2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 207–218. Springer Verlag, 2001.
- [17] Sergey Kitaev and Toufik Mansour. A survey on certain pattern problems. Preprint available at <http://www.ru.is/kennarar/sergey/publications.html>, 2003.
- [18] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading MA, 3rd edition, 1973.
- [19] Vincent Vatter. Enumeration schemes for restricted permutations. *Comb. Probab. Comput.*, 17(1):137–159, 2008.