



An algorithm for deciding the finiteness of the number of simple permutations in permutation classes

Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, Dominique Rossin

► To cite this version:

Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, Dominique Rossin. An algorithm for deciding the finiteness of the number of simple permutations in permutation classes. *Advances in Applied Mathematics*, 2015, 64, pp.124 - 200. 10.1016/j.aam.2014.12.001 . hal-01818308

HAL Id: hal-01818308

<https://hal.science/hal-01818308>

Submitted on 5 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An algorithm for deciding the finiteness of the number of simple permutations in permutation classes ¹

Frédérique Bassino^a, Mathilde Bouvel^{2,b}, Adeline Pierrot^c, Dominique Rossin^d

^a*Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, F-93430, Villetaneuse, France.*

^b*LaBRI UMR 5800, Univ. Bordeaux and CNRS, 351, cours de la Libération, 33405 Talence cedex, France; and Institut für Mathematik, Uni. Zürich, Zurich, Switzerland.*

^c*LIAFA UMR 7089, Univ. Paris Diderot and CNRS, Case 7014, 75205 Paris cedex 13, France; and Institut für Diskrete Mathematik und Geometrie, TU Wien, Vienna, Austria; and LRI UMR 8623, Univ. Paris Sud and CNRS, 91405 Orsay Cedex, France.*

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

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. This is a continuation of the work initiated in [Brignall, Ruškuc, Vatter, *Simple permutations: decidability and unavoidable substructures*, 2008], and shares several aspects with it. Like in this article, the main difficulty is to decide whether \mathcal{C} contains a finite number of proper pin-permutations, and this decision problem is solved using automata theory. Moreover, we use an encoding of proper pin-permutations by words over a finite alphabet, introduced by Brignall *et al.* However, unlike in their article, our construction of automata is fully algorithmic and efficient. It is based on the study of pin-permutations in [Bassino, Bouvel, Rossin, *Enumeration of pin-permutations*, 2011]. The complexity of the overall algorithm is $\mathcal{O}(n \log n + s^{2k})$ where n denotes the sum of the sizes of permutations in the basis B , s is the maximal size of a pin-permutation in B and k is the number of pin-permutations in B .

Keywords: Permutation class; finite basis; simple permutation; algorithm; automaton; pin-permutation.

Mathematics Subject Classification: 05A05; 68R05; 05-04.

1. Introduction

Since the definition of the pattern relation among permutations by Knuth in the 70's [17], the study of permutation patterns and permutation classes in com-

¹This work was completed with the support of the ANR (project ANR BLAN-0204_07 MAGNUM).

²Corresponding author. mathilde.bouvel@math.uzh.ch, +41 44 63 55 852 .

binatorics has been a quickly growing research field, and is now well-established. Most of the research done in this domain concerns *enumeration* questions on permutation classes. Another line of research on permutation classes has been emerging for about a decade: it is interested in properties or results that are less precise but apply to *families* of permutation classes. Examples of such general results may regard enumeration of permutation classes that fall into general frameworks, properties of the corresponding generating functions, growth rates of permutation classes, order-theoretic properties of permutation classes. . . This second point of view is not purely combinatorial but instead is intimately linked with algorithms. Indeed, 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 permutations has a finite basis and an algebraic generating function, and its developments by Brignall *et al.* in [9, 10, 11]. 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 of [3]: the permutations of the class (or rather their decomposition trees) can be described by a context-free grammar. In this theorem, as well as in other results obtained in this field, it appears that *simple permutations* play a crucial role. 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 towards algorithmic considerations. It takes its root in the theorem of Albert and Atkinson that we already mentioned, and follows its developments in [11] and [6].

In [11], 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. To this criterion, [11] associates 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 [11] 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 such permutations that do not belong to the class. This procedure is however not fully algorithmic, and its complexity is a double exponential, as we explain in Subsection 4.1.

Our goal is to solve the decision problem of [11] with an actual algorithm, whose complexity should be kept as low as possible. For this purpose, we heavily rely on [6] where we perform a detailed study of the class of *pin-permutations*, which contains the proper pin-permutations of [11]. These results allow us to

precisely characterize the pin words corresponding to any given pin-permutation, and to subsequently modify the automata construction of [11], leading to our algorithm deciding whether a permutation class given by a finite basis B contains a finite number of simple permutations. Figure 1 gives an overview of the general structure of our algorithm (the notations it uses will however be defined later, in Sections 3 and 4).

1. Check if \mathcal{C} contains finitely many parallel alternations and wedge simple permutations.
 2. Check if \mathcal{C} contains finitely many proper pin-permutations:
 - (a) Determine the set PB of pin-permutations of B ;
 - (b) For each π in PB , build an automaton \mathcal{A}_π recognizing the language $\overleftarrow{\mathcal{L}}_\pi$ (or a variant of this language);
 - (c) From the automata \mathcal{A}_π , build an automaton $\mathcal{A}_\mathcal{C}$ recognizing the language $\overleftarrow{\mathcal{M}} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$;
 - (d) Check if the language recognized by $\mathcal{A}_\mathcal{C}$ is finite.

Figure 1: Our algorithm testing if the number of simple permutations in $\mathcal{C} = Av(B)$ is finite.

As can be seen in Theorem 5.1 (p.24), the resulting algorithm is efficient: it 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 [11]. Notice that we described in [5] an algorithm solving the same problem on substitution-closed permutation classes, that is to say the classes of permutations whose bases contain only simple permutations. The complexity of our algorithm in this special framework is $\mathcal{O}(n \log n)$ where n is the sum of the size of the patterns in B .

The article is organized as follows. Section 2 starts with a reminder of previous definitions and results about permutation patterns, decomposition trees and pin-permutations. It also recalls from [11] the characterization of classes with a finite number of simple permutations, where proper pin-permutations enter into play. Section 3 establishes our criterion for deciding whether a permutation class contains a finite number of proper pin-permutations: this is the condition tested by the second step of the algorithm of Figure 1. Stating this criterion requires that we review the encoding of pin-permutations by pin words used by [11] and that we go further into the interpretation of the pattern order between pin-permutations in terms of words and languages. In Section 4, we describe and compare two methods for testing whether a class contains finitely many proper pin-permutations. We start with the procedure of [11], and proceed with our method. Then we outline in Subsection 4.3 the most technical part of our algorithm: building an automaton \mathcal{A}_π associated to every pin-permutation π of the basis of the class. Details and proofs for this step can be found in Appendices.

Finally, Section 5 describes and gives the complexity of our whole algorithm 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 6.

2. Preliminaries on permutations

We recall in this section a few definitions and results about permutation classes, substitution decomposition and decomposition trees, pin representations and pin-permutations. We also recall the characterization of classes with finitely many simple permutations. More details can be found in [3, 6, 9, 11].

2.1. Permutation classes and simple permutations

The topic of this paper is to answer algorithmically the question of whether a *permutation class* contains finitely many *simple permutations*, thereby ensuring that the generating function of the class is algebraic [3]. We naturally start by the definitions of this terminology.

A permutation $\sigma \in S_n$ is a bijective function from $\{1, \dots, n\}$ onto $\{1, \dots, n\}$. We represent a permutation σ either by the word $\sigma_1\sigma_2\dots\sigma_n$ where $\sigma_i = \sigma(i)$ for every $i \in \{1, \dots, n\}$, or by its *diagram* consisting in the set of points at coordinates (i, σ_i) drawn in the plane. Figure 3 (p.7) shows for example the diagram of $\sigma = 4726315$.

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 isomorphic to $\sigma_{i_1}\dots\sigma_{i_k}$ (see an example in Figure 3). 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 of permutations – *i.e.*, a set of permutations that are pairwise incomparable for \leq . The permutation class of *basis* B denoted $Av(B)$ is the set of all permutations avoiding every element of B .

The reader familiar with the permutation patterns literature will notice that we do not adopt the (equivalent) point of view of defining permutation classes as downward closed sets for \leq . Indeed, in this article, permutation classes are always given by their bases. We will further restrict our attention to classes having finite bases, since otherwise from [3], they contain infinitely many simple permutations.

A *block* (or *interval*) of a permutation σ of size n is a subset $\{i, \dots, (i+\ell-1)\}$ of consecutive integers of $\{1, \dots, n\}$ whose images under σ also form an interval of $\{1, \dots, n\}$. A permutation σ is *simple* when it is of size at least 4 and it contains no block, except the trivial ones: those of size 1 (the singletons) or of size n (σ itself). The only permutations of size smaller than 4 that have only trivial blocks are 1, 12 and 21, nevertheless they are *not* considered to be simple in this article.

2.2. Substitution decomposition and decomposition trees

Let σ be a permutation of S_k and π_1, \dots, π_k be k permutations of $S_{\ell_1}, \dots, S_{\ell_k}$ respectively. The *substitution* $\sigma[\pi_1, \pi_2, \dots, \pi_k]$ of $\pi_1, \pi_2, \dots, \pi_k$ in σ (also called *inflation* in [3]) is defined as the permutation whose diagram is obtained from the one of σ by replacing each point σ_i by a block containing the diagram of π_i . Alternatively, $\sigma[\pi_1, \pi_2, \dots, \pi_k]$ is the permutation of size $\sum \ell_i$ which is obtained as the concatenation $p_1 p_2 \dots p_k$ of sequences p_i of integers such that each p_i is isomorphic to π_i and all integers in p_i are smaller than those in p_j as soon as $\sigma_i < \sigma_j$. For example $132[21, 132, 1] = 214653$.

Permutations can be decomposed using substitutions, as described in Theorem 2.2 below. For this purpose, we now introduce some definitions and notations. For any $k \geq 2$, let I_k be the permutation $12\dots k$ and D_k be $k(k-1)\dots 1$. Denote by \oplus and \ominus respectively I_k and D_k . Notice that in inflations of the form $\oplus[\pi_1, \pi_2, \dots, \pi_k] = I_k[\pi_1, \pi_2, \dots, \pi_k]$ or $\ominus[\pi_1, \pi_2, \dots, \pi_k] = D_k[\pi_1, \pi_2, \dots, \pi_k]$, the integer k is determined without ambiguity by the number of permutations π_i of the inflation.

Definition 2.1. A permutation σ is \oplus -decomposable (resp. \ominus -decomposable) if it can be written as $\oplus[\pi_1, \pi_2, \dots, \pi_k]$ (resp. $\ominus[\pi_1, \pi_2, \dots, \pi_k]$), for some $k \geq 2$. Otherwise, it is \oplus -indecomposable (resp. \ominus -indecomposable).

Theorem 2.2. For any $n \geq 2$, every permutation $\sigma \in S_n$ can be uniquely decomposed as either:

- $\oplus[\pi_1, \pi_2, \dots, \pi_k]$, with $k \geq 2$ and $\pi_1, \pi_2, \dots, \pi_k$ \oplus -indecomposable,
- $\ominus[\pi_1, \pi_2, \dots, \pi_k]$, with $k \geq 2$ and $\pi_1, \pi_2, \dots, \pi_k$ \ominus -indecomposable,
- $\alpha[\pi_1, \dots, \pi_k]$ with α a simple permutation and $k = |\alpha|$ (so that $k \geq 4$).

Theorem 2.2 appears in [3] under a form that is trivially equivalent. The reader can also refer to [14] for a historical reference, or to [15] for a reference in an algorithmic context.

Remark 2.3. 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 .

Theorem 2.2 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. This complete decomposition is called the *substitution decomposition* of a permutation. It is accounted for by a tree, called the *substitution decomposition tree*, where a substitution $\alpha[\pi_1, \dots, \pi_k]$ is represented by a node labeled α with k ordered children representing the π_i .

Definition 2.4. 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 $|\alpha|$ children. See Figure 2 for an example.

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 are \oplus -indecomposable (resp. \ominus -indecomposable) in the first (resp. second) item of Theorem 2.2.

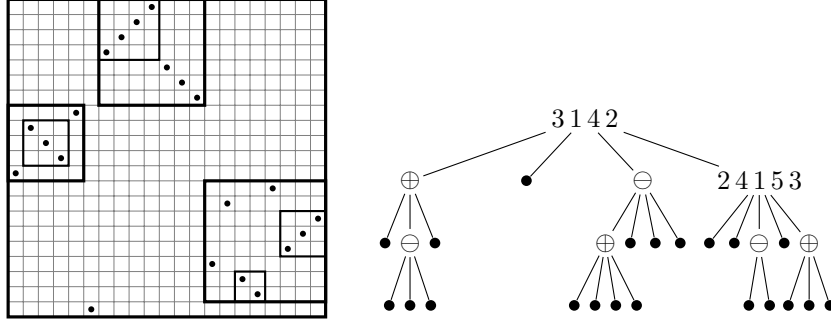


Figure 2: The diagram and the substitution decomposition tree T of the permutation $\sigma = 10\ 13\ 12\ 11\ 14\ 1\ 18\ 19\ 20\ 21\ 17\ 16\ 15\ 4\ 8\ 3\ 2\ 9\ 5\ 6\ 7$. The internal nodes of T correspond to the blocks of σ marked by rectangles.

Remark 2.5. *Permutations are bijectively characterized by their substitution decomposition trees.*

In the sequel, when writing *a child of a node V* we mean the permutation corresponding to the subtree rooted at this child of node V .

2.3. Pin-permutations and pin representations

In this article, the *pin-permutations* (and their decomposition trees) play a central role. The remaining of this preliminary section recalls their definition and explains how they are related to our problem of testing whether a permutation class contains finitely many simple permutations.

A *pin* is a point in the plane. 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 containing P and the other one containing Q . The *bounding box* (also known as the *rectangular hull*) of a set of points P is the smallest axis-parallel rectangle containing the set P . A *pin sequence* is a sequence (p_1, \dots, p_k) of pins in the plane such that no two points are horizontally or vertically aligned and for all $i \geq 2$, p_i lies outside the bounding box of $\{p_1, \dots, p_{i-1}\}$ and satisfies one of the following conditions:

- *separation condition*: p_i separates p_{i-1} from $\{p_1, \dots, p_{i-2}\}$;
- *independence condition*: 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 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 3). Not all permutations are pin-permutations (see for example the permutation σ of Figure 3).

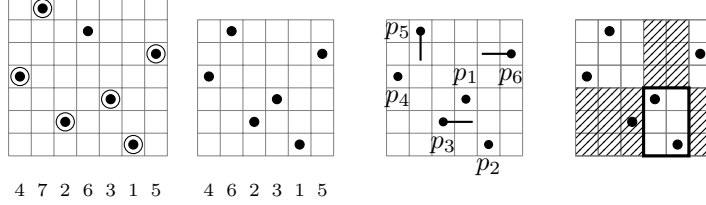


Figure 3: The permutation $\sigma = 4726315$, its pattern $\pi = 462315$, a pin representation p of π , and the bounding box of $\{p_1, p_2\}$ with its sides shaded.

Lemma 2.17 of [6] is used several times in our proofs, and we state it here:

Lemma 2.6. *Let (p_1, \dots, p_n) be a pin representation of $\sigma \in S_n$. Then for each $i \in \{2, \dots, n-1\}$, if there exists a point x of σ on the sides of the bounding box of $\{p_1, \dots, p_i\}$, then it is unique and $x = p_{i+1}$.*

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 2.7. *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 $\{p_1, \dots, p_{i-1}\}$.*

2.4. Characterization of classes with finitely many simple permutations

In [11], Brignall *et al.* provide a criterion characterizing when a class contains a finite number of simple permutations. They show that it is equivalent to the class containing a finite number of permutations of three simpler kinds, which they define. Among the three new kinds of permutations that they introduce are the proper pin-permutations that we have already seen, but also the *parallel alternations* and the *wedge simple permutations*. The definition of these families of permutations is not crucial to our work, hence we refer the reader to [11] for more details, and to Figure 4 for examples.

Theorem 2.8. [9, 11] *A permutation class $\text{Av}(B)$ contains a finite number of simple permutations if and only if it contains:*

- a finite number of wedge simple permutations, and
- a finite number of parallel alternations, and
- a finite number of proper pin-permutations.

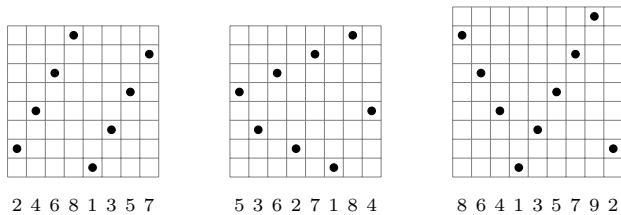


Figure 4: From left to right: a parallel alternation, and two wedge simple permutations (of type 1 and 2 respectively).

Notice also that in Theorem 2.8 above, the proper pin sequences of [11] have been replaced by proper pin-permutations. But containing a finite number of proper pin-permutations is equivalent to containing a finite number of proper pin sequences. Indeed, the encoding of proper pin-permutations by proper pin sequences provides a finite-to-one correspondence.

Whereas the exact definitions of the wedge simple permutations and the parallel alternations have been omitted here, it is however essential for our purpose to be able to test whether a class given by a finite basis contains a finite number of parallel alternations and wedge simple permutations. Parallel alternations and wedge simple permutations, that can be of type 1 or 2, are well characterized in [11]. This characterization leads to the following lemmas:

Lemma 2.9. [11] *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 2.10. [11] *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 2.11. [11] *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 $Av(2134, 2143, 3124, 3142, 3241, 3412, 4123, 4132, 4231, 4312)$.*

Using these lemmas together with a result of [2] we have:

Lemma 2.12. *Testing whether a finitely based class $Av(B)$ contains a finite number of wedge simple permutations and parallel alternations can be done in $\mathcal{O}(n \log n)$ time, where $n = \sum_{\pi \in B} |\pi|$.*

Proof. From Lemmas 2.9 to 2.11, deciding if $Av(B)$ contains a finite number of wedge simple permutations and parallel alternations is equivalent to checking if elements of its basis B involve patterns of size at most 4. From [2] checking whether a permutation π involves a fixed set of patterns of size at most 4 can be done in $\mathcal{O}(|\pi| \log |\pi|)$. As we check for each permutation of B the involvement of fixed sets of permutations of size at most 4, this leads to a $\mathcal{O}(n \log n)$ algorithm

for deciding whether the number of parallel alternations and of wedge simple permutations in the class is finite. \square

In [11] Brignall *et al.* also proved that it is decidable whether $\mathcal{C} = Av(B)$ contains a finite number of proper pin-permutations. Their proof heavily relies on an encoding of proper pin-permutations by words over a finite alphabet (called *pin words*), and on language theoretic arguments. In the next section, we review and further develop the theory of pin words. Then, in Section 4 we will review the decision procedure of [11], and explain how we could modify it into an efficient algorithm.

3. Characterization of classes with finitely many proper pin-permutations

Our goal in this section is to provide a criterion (that can be tested algorithmically, in the next sections) for a permutation class $\mathcal{C} = Av(B)$ given by its finite basis B to contain finitely many proper pin-permutations. The encoding of pin-permutations by their pin words – to be reviewed in Subsection 3.1 – has an essential property that can be used in establishing such a criterion: it allows to interpret the pattern order \leq on pin-permutations as an order relation \preceq on their pin words.

This property is already at the core of [11], and we recall it below as Lemma 3.8. In [11], it is used to derive a first criterion on \mathcal{C} to contain a finite number of proper pin-permutations:

\mathcal{C} contains finitely many proper pin-permutations if and only if the language $\mathcal{SP} \setminus \bigcup \{\text{strict pin word } w \mid u \preceq w\}$ is finite, where the union is taken over all pin words u that encode a permutation $\pi \in B$ and \mathcal{SP} denote the language of all strict pin words (see Definition 3.4).

This criterion may then be decided using automata theory, as explained in [11] and reviewed in Subsection 4.1.

In what follows, we go further into the encoding of pin-permutations by words, and into the interpretation of the pattern order \leq in terms of words and languages. This allows us to associate a language \mathcal{L}_π to every pin-permutation π in such a way that if $\pi \leq \sigma$ then $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$. Subsequently, these languages \mathcal{L}_π can be used to characterize when \mathcal{C} contains a finite number of proper pin-permutations – see Theorem 3.18:

\mathcal{C} contains finitely many proper pin-permutations if and only if the language $\mathcal{M} \setminus \bigcup \mathcal{L}_\pi$ is finite, where the union is taken over all pin-permutations $\pi \in B$ and \mathcal{M} denotes the set of words on the alphabet $\{L, R, U, D\}$ with no factor in $\{UU, UD, DU, DD, LL, LR, RL, RR\}$ (see p.11).

As we shall see in Section 4, this new criterion can be tested by an algorithm, far more efficiently than the first criterion above.

3.1. Pin words

Pin representations can be encoded on the alphabet $\{1, 2, 3, 4, U, D, L, R\}$ by words 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 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 p_i is independent from $\{p_0, p_1, \dots, p_{i-1}\}$ and is located 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 5. The region encoded by 1 is called the first *quadrant* with respect to the box \blacksquare . The same goes for 2, 3, 4. The letters U, D, L, R are called *directions*, while 1, 2, 3 and 4 are *numerals*.

2	U	1
L	\blacksquare	R
3	D	4

Figure 5: Encoding of pins by letters.

4R	3R	p_2
41	31	3U
11	21	2U

Figure 6: 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.

Example 3.1. $14L2UR$ (if p_0 is between p_3 and p_1) and $3DL2UR$ (if p_0 is horizontally between p_1 and p_4 and vertically between p_2 and p_6) are pin words corresponding to the pin representation of $\pi = 462315$ shown in Figure 3 (p.7).

Example 3.1 shows in particular that several pin words encode the same pin representation, depending on the choice of the origin p_0 . We may actually describe the number of these pin words:

Remark 3.2. *Because of the choice of the origin p_0 , each pin-permutation of size greater than 1 has at least 6 pin words. More precisely each pin representation p is encoded by 6 pin words if p_3 is a separating pin and 8 pin words otherwise (see Figure 6). Indeed, once a pin representation p is fixed, the letters encoding p_i for $i \geq 3$ in a pin word encoding p are uniquely determined.*

Conversely, pin words indeed encode pin-permutations since to each pin word corresponds a unique pin representation, hence a unique permutation.

Remark 3.3. *The definition of pin sequences implies that pin words do not contain any of the factors $UU, UD, DU, DD, LL, LR, RL$ and RR .*

Definition 3.4. A strict (resp. quasi-strict) pin word is a pin word that begins with a numeral (resp. two numerals) followed only by directions. We denote by \mathcal{SP} the set of all strict pin words.

Remark 3.5 (Proper pin representations, strict and quasi-strict pin words).

Every pin word encoding a proper pin representation is either strict or quasi-strict. Conversely if a pin word is strict or quasi-strict, then the pin representation it encodes is proper. Finally a pin-permutation is proper if and only if it admits a strict pin word.

3.2. Pattern containment and piecewise factor relation

Recall the definition of the partial order \preceq on pin words introduced in [11].

Definition 3.6. 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 non-empty, the first letter of $w^{(i)}$ corresponds to a point lying in the quadrant – w.r.t. the origin of the encoding w – specified by the first letter of $u^{(i)}$, and all other letters in $u^{(i)}$ and $w^{(i)}$ agree.

Example 3.7. The strong numeral-led factor decomposition of $u = 14L2UR$ is $u = 1 \cdot 4L \cdot 2UR$. Moreover, $u \preceq w = 2RU4LULURD4L$, because w may be decomposed as $w = 2RU \cdot \mathbf{4L} \cdot \mathbf{ULUR} \cdot D4L$, where the factors $w^{(i)}$ satisfying the conditions of Definition 3.6 are emphasized by bold letters.

As we mentioned already, the essential property of this order is that it is closely related to the pattern containment order \leq on permutations.

Lemma 3.8. [11] If the pin word w encodes the permutation σ and $\pi \leq \sigma$ then there is a pin word u encoding π with $u \preceq w$. Conversely if $u \preceq w$ then the permutation corresponding to u is contained in the one corresponding to 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 . Our purpose in the following is to adapt the relation \preceq into an actual piecewise factor relation. This is achieved in Theorem 3.13, using a further encoding of pin words that we introduced in [5] and recall hereafter.

Recall that \mathcal{SP} denotes the set of strict pin words, and let $\mathcal{SP}_{\geq 2} = \mathcal{SP} \setminus \{1, 2, 3, 4\}$ be the set of strict pin words of length at least 2. Further denote by \mathcal{M} (resp. $\mathcal{M}_{\geq 3}$) the set of words of length at least 2 (resp. at least 3) over the alphabet $\{L, R, U, D\}$ such that L, R is followed by U, D and conversely. We define below a bijection that sends strict pin words to words of \mathcal{M} . It consists

of replacing the only numeral in a strict pin word by two directions. Intuitively, given a numeral q and a box \blacksquare , inserting two pins in the two directions prescribed by the bijection ends up in a pin lying in quadrant q with respect to the box \blacksquare .

Definition 3.9. We define a bijection ϕ from $\mathcal{SP}_{\geq 2}$ to $\mathcal{M}_{\geq 3}$ as follows. For any strict pin word $u \in \mathcal{SP}_{\geq 2}$ 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 $u = u_1u_2 \dots \in \mathcal{SP}_{\geq 2}$, $u_i = \phi(u)_{i+1}$ for any $i \geq 2$.

In the above table, we can notice that, for any $u \in \mathcal{SP}_{\geq 2}$, 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 \mathcal{SP} by setting for words of length 1: $\phi(1) = \{UR, RU\}$, $\phi(2) = \{UL, LU\}$, $\phi(3) = \{DL, LD\}$ and $\phi(4) = \{RD, DR\}$, and by defining consistently $\phi^{-1}(v) \in \{1, 2, 3, 4\}$ for any v in $\{LU, LD, RU, RD, UL, UR, DL, DR\}$.

Lemma 3.10 below shows that for each pin word w , we know in which quadrant (w.r.t. the origin of the encoding) lies every pin of the pin representation p corresponding to w . More precisely for each $i \leq |w|$, knowing only w_i and w_{i-1} , we can determine in which quadrant p_i lies.

Lemma 3.10. Let w be a pin word and p be the pin representation corresponding to w . For any $i \geq 2$, the numeral indicating the quadrant in which p_i lies with respect to $\{p_0, \dots, p_{i-2}\}$ is

$$\begin{cases} w_i & \text{if } w_i \text{ is a numeral;} \\ \phi^{-1}(w_{i-1}w_i) & \text{if } w_{i-1} \text{ and } w_i \text{ are directions;} \\ \phi^{-1}(BC) & \text{otherwise, with } \phi(w_{i-1}w_i) = ABC. \end{cases}$$

Notice that in the third case w_{i-1} is a numeral and w_i is a direction; consequently, $ABC \in \mathcal{M}_3$ is given by the table of Definition 3.9.

Proof. Similar to the proof of Lemma 3.4 of [5], adapted to the case where w is any pin word, i.e., is not necessarily strict. \square

Lemma 3.10 is used in the proofs of Lemma 3.12 and Theorem 3.13. Their statement also requires that we extend some definitions from \mathcal{SP} to \mathcal{M} .

Remark 3.11. Words of \mathcal{M} may also be seen as encodings of pin sequences (as in Subsection 3.1), taking the origin p_0 to be a box instead of a point. Moreover, the relation $u \preceq w$ can be extended to $w \in \mathcal{M}$, and the map ϕ can be defined on words of \mathcal{M} as the identity map (although this extension of ϕ to the domain $\mathcal{SP} \cup \mathcal{M}$ is not a bijection anymore).

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.

Lemma 3.12. *Let u be a strict pin word and w be a word of $\mathcal{SP} \cup \mathcal{M}$. If $|u| \geq 2$ then $u \preceq w$ if and only if $\phi(u)$ is a factor of $\phi(w)$. If $|u| = 1$ then $u \preceq w$ if and only if $\phi(w)$ has a factor in $\phi(u)$.*

Proof. Note that if $|u| \geq 2$ then $\phi(u)$ is a word but if $|u| = 1$ then $\phi(u)$ is a set of two words. The case where $|u| \geq 2$ and w is a strict pin word corresponds exactly to Lemma 3.5 of [5]. Other cases are proved in a similar way, using, instead of Lemma 3.4 of [5], its generalization provided by our current Lemma 3.10. \square

In the statement of Lemma 3.12, we have distinguished the cases $|u| \geq 2$ and $|u| = 1$ since $\phi(u)$ is a word or a set of two words in these respective cases. However, to avoid such uselessly heavy statements, we do not make this distinction in the sequel, and we write indifferently “ $\phi(u)$ is a factor of w ” or

“ w has a factor in $\phi(u)$ ” meaning that $\begin{cases} \text{if } |u| = 1, w \text{ has a factor in } \phi(u) \\ \text{if } |u| \geq 2, \phi(u) \text{ is a factor of } w. \end{cases}$

When the pin word u is not strict, Lemma 3.12 can be extended formalizing the idea of piecewise factors mentioned at the beginning of this section.

Theorem 3.13. *Let u and w be two pin words and $u = u^{(1)} \dots u^{(j)}$ be the strong numeral-led factors decomposition of u . Then $u \preceq w$ if and only 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 $i \in \{1, \dots, j\}$, $w^{(i)} \in \mathcal{SP} \cup \mathcal{M}$ and $\phi(w^{(i)})$ has a factor in $\phi(u^{(i)})$.*

Proof. We prove that $u \preceq w$ if and only 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 $i \in \{1, \dots, j\}$, $w^{(i)} \in \mathcal{SP} \cup \mathcal{M}$ and $u^{(i)} \preceq w^{(i)}$. Then the result follows using Lemma 3.12.

If $u \preceq w$, then w can be chopped into $w = \bar{v}^{(1)}\bar{w}^{(1)} \dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$ as in Definition 3.6. We set $w^{(i)} = \bar{w}^{(i)}$ if $\bar{w}^{(i)}$ begins with a numeral, and we take $w^{(i)}$ to be the suffix of $\bar{v}^{(i)}\bar{w}^{(i)}$ of length $|\bar{w}^{(i)}| + 1$ otherwise. Then for all $i \in \{1, \dots, j\}$, $w^{(i)} \in \mathcal{SP} \cup \mathcal{M}$ and we have $u^{(i)} \preceq w^{(i)}$. Indeed, if $\bar{w}^{(i)}$ begins with a direction, by Lemma 3.10 the point corresponding to the first letter of $\bar{w}^{(i)}$ lies in the quadrant determined by the last letter of $\bar{v}^{(i)}$ and the first letter of $\bar{w}^{(i)}$ (w.r.t. the origin of the encoding w and also of the encoding $w^{(i)}$).

Conversely if w can be chopped into $w = v^{(1)}w^{(1)} \dots v^{(j)}w^{(j)}v^{(j+1)}$ such that for all $i \in \{1, \dots, j\}$, $u^{(i)} \preceq w^{(i)}$ then from Definition 3.6 we can decompose $w^{(i)}$ as $y^{(i)}\bar{w}^{(i)}z^{(i)}$ and thanks to Lemma 3.10 it is sufficient to set $\bar{v}^{(i)} = z^{(i-1)}v^{(i)}y^{(i)}$ to have $w = \bar{v}^{(1)}\bar{w}^{(1)} \dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$ as in Definition 3.6. \square

3.3. Pattern containment and set inclusion

Recall that our goal is to characterize when there are finitely many proper pin-permutations in a class. By Remark 3.5, for any proper pin-permutation σ , there exists a **strict** pin word w_σ that encodes σ . And from Lemma 3.8, 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 $u \preceq w_\sigma$. Consequently, to study the proper pin-permutations not in $\mathcal{C} = Av(B)$, *i.e.* those containing some pattern π in B , it is enough to study the **strict** pin words containing some pin word u encoding π , for π in B . This is the reason why we introduce the languages $\mathcal{L}(u)$ and \mathcal{L}_π below.

Definition 3.14. *Let u be a pin word and $u = u^{(1)} \dots u^{(j)}$ be its strong numeral-led factor decomposition. We set*

$$\mathcal{L}(u) = A^* \phi(u^{(1)}) A^* \phi(u^{(2)}) \dots A^* \phi(u^{(j)}) A^* \text{ where } A = \{U, D, L, R\}.$$

Let π be a permutation, and $P(\pi)$ be the set of pin words that encode π . We set

$$\mathcal{L}_\pi = \cup_{u \in P(\pi)} \mathcal{L}(u)$$

As we shall see in Lemma 3.17, the languages \mathcal{L}_π allow to describe the strict pin words of proper pin-permutations that contain π (or rather their image by ϕ). Note however that not all words of \mathcal{L}_π are in the image by ϕ of the strict pin words of proper pin-permutations containing π . For instance, there are words starting with $LLLLL\phi(u^{(1)})$ that belong to \mathcal{L}_π , and these are not even in \mathcal{M} (*i.e.*, are not the image of a strict pin word by ϕ). But Lemma 3.17 proves that such trivial “bad words” not belonging to \mathcal{M} are the only ones that we should exclude from \mathcal{L}_π . Indeed $\mathcal{L}_\pi \cap \mathcal{M}$ is in one-to-one correspondence with strict pin words encoding proper pin-permutations that contain π , via ϕ^{-1} .

These languages \mathcal{L}_π further have the interesting property of somehow translating the pattern involvement between pin-permutations into set inclusion, as expressed by Theorem 3.15.

Note that \mathcal{L}_π is non-empty if and only if $P(\pi)$ is non-empty or equivalently π is a pin-permutation. So when π is not a pin-permutation, the results of Theorem 3.15 and Lemma 3.17 follow easily from the following statement (see for instance Lemma 3.3 of [6]): if $\pi \leq \sigma$ and σ is a pin-permutation, then π is a pin-permutation.

Theorem 3.15. *Let π and σ be permutations, such that $\pi \leq \sigma$. Then $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$.*

In the following, we write $m = v^{(1)}\phi(u^{(1)})v^{(2)}\phi(u^{(2)}) \dots v^{(j)}\phi(u^{(j)})v^{(j+1)}$ for $m \in A^*$, meaning that $m = v^{(1)}w^{(1)}v^{(2)}w^{(2)} \dots v^{(j)}w^{(j)}v^{(j+1)}$ with $w^{(i)} \in \phi(u^{(i)})$ if $u^{(i)}$ has length 1 and $w^{(i)} = \phi(u^{(i)})$ otherwise.

Proof. Suppose that $\pi \leq \sigma$. If \mathcal{L}_σ is empty, the statement trivially holds. Otherwise, let $m \in \mathcal{L}_\sigma$. We want to show that $m \in \mathcal{L}_\pi$. By definition of \mathcal{L}_σ , there exists $w \in P(\sigma)$ such that $m = v^{(1)}\phi(w^{(1)})v^{(2)} \dots \phi(w^{(i)})v^{(i+1)}$ where $w = w^{(1)}w^{(2)} \dots w^{(i)}$ is the strong numeral-led factor decomposition of w . From Lemma 3.8 there is $u \in P(\pi)$ such that $u \preceq w$. Let $u = u^{(1)} \dots u^{(j)}$ be the strong numeral-led factor decomposition of u . From Theorem 3.13, $w = \bar{v}^{(1)}\bar{w}^{(1)} \dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$ where for all $k \in \{1, \dots, j\}$, $\bar{w}^{(k)} \in \mathcal{SP} \cup \mathcal{M}$

and $\phi(\bar{w}^{(k)})$ has a factor in $\phi(u^{(k)})$. But $w^{(1)}w^{(2)}\dots w^{(i)}$ is the strong numeralled factor decomposition of $w = \bar{v}^{(1)}\bar{w}^{(1)}\dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$. Therefore the factors $\bar{w}^{(1)}, \bar{w}^{(2)}, \dots, \bar{w}^{(j)}$ appear in this order in $w^{(1)}w^{(2)}\dots w^{(i)}$, being non-overlapping and each inside one $w^{(\ell)}$, since each $w^{(\ell)}$ begins with a numeral and each $\bar{w}^{(k)}$ is in $\mathcal{SP} \cup \mathcal{M}$. Thus by definition of ϕ , the factors $\phi(\bar{w}^{(1)}), \phi(\bar{w}^{(2)}), \dots, \phi(\bar{w}^{(j)})$ appear in this order in $\phi(w^{(1)})\phi(w^{(2)})\dots \phi(w^{(i)})$, being non-overlapping and each inside one $\phi(w^{(\ell)})$. So $m = v^{(1)}\phi(w^{(1)})v^{(2)}\phi(w^{(2)})\dots v^{(i)}\phi(w^{(i)})v^{(i+1)} \in A^*\phi(\bar{w}^{(1)})A^*\phi(\bar{w}^{(2)})\dots A^*\phi(\bar{w}^{(j)})A^*$. But for all $k \in \{1, \dots, j\}$, $\phi(\bar{w}^{(k)})$ has a factor in $\phi(u^{(k)})$, thus $m \in \mathcal{L}(u) = A^*\phi(u^{(1)})A^*\phi(u^{(2)})\dots A^*\phi(u^{(j)})A^*$ and so $m \in \mathcal{L}_\pi$. \square

Remark 3.16. *Although it is not necessary for our purpose, we would find interesting to have a stronger version of Theorem 3.15 which would state the equivalence between $\pi \leq \sigma$ and $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$, when π and σ are pin-permutations. We do not know if this equivalence holds. However, we do know that, with a small modification of $\mathcal{L}(u)$ to allow for an extra symbol in A (which plays the role of a separator), then we have, for all pin-permutations π and σ , $\pi \leq \sigma$ if and only if $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$.*

3.4. Characterizing when a class has a finite number of proper pin-permutations

We conclude Section 3 by putting together the above definitions and results to answer to our original problem: providing a criterion that characterizes when a permutation class contains finitely many proper pin-permutations.

Lemma 3.17. *Let σ be a proper pin-permutation, π be a permutation and w be a strict pin word encoding σ . Then $\pi \leq \sigma$ if and only if $\phi(w) \in \mathcal{L}_\pi$.*

Proof. Assume that $\pi \leq \sigma$, then from Theorem 3.15 $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$. As w is a strict pin word, $\phi(w) \in \mathcal{L}(w)$ thus $\phi(w) \in \mathcal{L}_\sigma$ and so $\phi(w) \in \mathcal{L}_\pi$.

Conversely, assume that $\phi(w) \in \mathcal{L}_\pi$. Then there exists a pin word u encoding π such that $\phi(w) \in \mathcal{L}(u)$. Let us denote by $u = u^{(1)}\dots u^{(j)}$ the strong numeralled factor decomposition of u . By definition of $\mathcal{L}(u)$, $\phi(w)$ can be decomposed into $t^{(1)}\dots t^{(j+1)}$, with $t^{(i)} \in A^*\phi(u^{(i)}) \cap \mathcal{M}$ for $i \in \{1, \dots, j\}$. By definition of ϕ and since w is a strict pin word, there exists a strict pin word t such that $w = t t^{(2)}\dots t^{(j+1)}$. Then $\phi(t) = t^{(1)}$ and $\phi(u^{(1)})$ is a factor of $\phi(t)$. Furthermore, for $i \in \{2, \dots, j\}$, $\phi(u^{(i)})$ is a factor of $\phi(t^{(i)}) = t^{(i)}$ (this equality holds because $t^{(i)} \in \mathcal{M}$). Consequently, from Theorem 3.13, $u \preccurlyeq w$. Finally from Lemma 3.8, we conclude that $\pi \leq \sigma$. \square

By ϕ^{-1} , each word of \mathcal{M} is turned into a strict pin word and hence into a proper pin-permutation. As a consequence of Lemma 3.17, $\mathcal{L}_\pi \cap \mathcal{M}$ is the image by ϕ of the language of strict pin words encoding proper pin-permutations σ that contain π as a pattern: $\mathcal{L}_\pi \cap \mathcal{M} = \{\phi(w) \mid \exists \sigma \text{ such that } \pi \leq \sigma \text{ and } w \in \mathcal{SP} \cap P(\sigma)\}$. With the same idea we have the following theorem, which provides the criterion announced at the beginning of Section 3:

Theorem 3.18. *A permutation class $Av(B)$ contains a finite number of proper pin-permutations if and only if the set $\mathcal{M} \setminus \bigcup_{\pi \in B} \mathcal{L}_\pi$ is finite.*

Proof. Let S_B be the set of strict pin words encoding permutations of size at least 2 in $Av(B)$. Then ϕ is a bijection from S_B to $\mathcal{M}_{\geq 3} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$. Indeed for any strict pin word w of length at least 2, let σ be the permutation encoded by w . Then σ is a proper pin-permutation and Lemma 3.17 implies that $\sigma \in Av(B)$ if and only if $\phi(w) \notin \cup_{\pi \in B} \mathcal{L}_\pi$. We conclude the proof observing that every proper pin-permutation σ of size n is associated to at least 1 and (very loosely) at most 8^n strict pin words, as any pin word encoding σ is a word of length n over an 8-letter alphabet. \square

4. Algorithm(s) testing if a class contains a finite number of proper pin-permutations

Section 3 (and specifically Theorem 3.18) provides us with a characterization of classes $Av(B)$ which contain finitely many proper pin-permutations: they are those such that $\mathcal{M} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$ is finite. Our goal is now to find an algorithm, as efficient as possible, checking this condition.

The general structure of this algorithm will be explained in Subsection 4.2. It involves the use of some automata \mathcal{A}_π recognizing \mathcal{L}_π for any pin-permutation $\pi \in B$. The construction of these automata is rather technical. Subsection 4.3 will give an overview of it, while the technical details are postponed to Appendices B and C.

Before we get to our algorithm, we first review the decision procedure of [11], which also answers the question of whether a class $Av(B)$ contains finitely many proper pin-permutations. This review will serve two purposes: one is to help the reader see the common aspects and the differences between this procedure and our algorithm; the other is to be able to compare their complexities.

4.1. The decision procedure of Brignall, Ruškuc and Vatter

As reviewed at the beginning of Section 3, Brignall *et al.* provided in [11] a first characterization of classes containing finitely many proper pin-permutations. Namely, they proved that $\mathcal{C} = Av(B)$ contains a finite number of proper pin-permutations if and only if the set $\mathcal{L} = \mathcal{SP} \setminus \bigcup_{u \in P(B)} \{\text{strict pin word } w \mid u \preceq w\}$ is finite, where $P(B)$ denotes the set of pin words encoding a permutation of B . The main point of the procedure of [11] (which will be similar in our algorithm) is then to decide the finiteness of the language \mathcal{L} using automata theory.

Given a pin word u , the authors of [11] explain how to build an automaton $\mathcal{A}^{(u)}$ recognizing a language $\mathcal{L}^{(u)}$ such that $\mathcal{SP} \cap \mathcal{L}^{(u)} = \{\text{strict pin word } w \mid u \preceq w\}$. Then, they notice that \mathcal{SP} is a recognizable language, and conclude – with classical theorems of automata theory – that it is decidable whether the language \mathcal{L} is finite, *i.e.*, whether \mathcal{C} contains a finite number of proper pin-permutations.

The proof of [11] is constructive and establishes that deciding whether \mathcal{C} contains a finite number of proper pin-permutations *may* be done algorithmically. However the authors do not give an actual algorithm since many steps are not given explicitly. More precisely, if we turn into an actual algorithm the procedure of [11], the main steps would be:

1. Compute the set $P(B)$ of pin words encoding permutations of B ;
2. For each $u \in P(B)$, build the automaton $\mathcal{A}^{(u)}$ recognizing $\mathcal{L}^{(u)}$;
3. Build an automaton \mathcal{A} recognizing $\mathcal{L} = \mathcal{SP} \setminus \bigcup_{u \in P(B)} \mathcal{L}^{(u)}$;
4. Test whether the language accepted by \mathcal{A} is finite.

In [11], the authors focus on the second step (which is indeed the main one), even though the complexity of building $\mathcal{A}^{(u)}$ is not analyzed. The first step is not addressed in [11], and the third (resp. fourth) step is solved applying an existential (resp. decidability) theorem of automata theory – in particular, the complexity of the corresponding construction (resp. decision) is not studied. Analyzing the above four-step procedure, we prove in the following that it has a doubly exponential complexity due to the resolution of a co-finiteness problem for a regular language given by a non-deterministic automaton. Let us first introduce some notations: denote by n the sum of the sizes of permutations in the basis B , by s' (resp. s) the maximal size of a permutation (resp. *pin*-permutation) in B and by k the number of pin-permutations in B .

Even though [11] does not study the first step of the above procedure, there is a naive algorithm to solve it: for each permutation π in B , for each pin word u of length $|\pi|$, check if the permutation encoded by u is π , and add u to $P(B)$ in the affirmative. This is performed in $\mathcal{O}(n \cdot 8^{s'})$ time. In our work we explain how to replace this step by a step solved in $\mathcal{O}(n)$ time.

For the second step, [11] explains how to build automata $\mathcal{A}^{(u)}$ recognizing the languages $\mathcal{L}^{(u)}$. We will not detail the analysis of the complexity of building $\mathcal{A}^{(u)}$, but let us notice that these automata are non deterministic and have $\mathcal{O}(|u|)$ (and at least $|u|$) states.

About the third step, [11] refers to automata theory without detail. The most direct way to achieve this step is to build by juxtaposition an intermediate automaton $\mathcal{A}^{P(B)}$ recognizing $\bigcup_{u \in P(B)} \mathcal{L}^{(u)}$, to determinize this automaton in

order to complement it, and then to compute the intersection with an automaton recognizing \mathcal{SP} . But the determinization of an automaton is exponential w.r.t. the number of states of the automaton. Since the number of states of $\mathcal{A}^{P(B)}$ is $\mathcal{O}(\sum_{u \in P(B)} |u|)$ and is indeed at least $\sum_{u \in P(B)} |u|$, the complexity of this algorithm for the third step is $\mathcal{O}(2^{\sum_{u \in P(B)} |u|})$. Moreover, $\sum_{u \in P(B)} |u| \leq k \cdot 8^s \cdot s$. Even if this bound may not be tight, there exist pin-permutations encoded by an exponential number of pin words. For instance, the identity of size s is encoded by at least 2^s pin words, since any word on the alphabet $\{1, 3\}$ is suitable. Therefore the complexity of the above algorithm for the third step is of order at least $\mathcal{O}(2^{k \cdot s \cdot 2^s})$, which is doubly exponential w.r.t. s .

Finally, [11] refers to a classical algorithm for the fourth step. It consists in testing whether the automaton \mathcal{A} obtained at the end of the third step contains a cycle that can be reached from an initial state and can lead to a final state. This is linear w.r.t. the size of the automaton \mathcal{A} (and we will detail why in Subsection 5.3, as our algorithm ends with a similar step).

4.2. A more efficient alternative

Reviewing the procedure of [11] and analyzing its complexity, we have seen why this procedure is not efficient. The main issue is the determinization of the automaton $\mathcal{A}^{P(B)}$ recognizing $\bigcup_{u \in P(B)} \mathcal{L}^{(u)}$ (so that it may be complemented).

A secondary issue is that this automaton $\mathcal{A}^{P(B)}$ is built by juxtaposition of a large number of automata: one for each pin word in $P(B)$. In this subsection, we explain how to overcome these two issues.

First, we do not start from the same characterization of classes $\mathcal{C} = Av(B)$ containing a finite number of proper pin-permutations. Instead of the one of [11], we take our alternative criterion provided by Theorem 3.18, and provide an algorithm testing whether the language $\mathcal{M} \setminus \bigcup_{\pi \in B} \mathcal{L}_\pi$ is finite.

In this second characterization, the languages \mathcal{L}_π play the same role as the languages $\mathcal{L}^{(u)}$ in the first one. There is however only one language for each $\pi \in B$, that somehow accounts for all the languages $\mathcal{L}^{(u)}$ for $u \in P(\pi)$. This solves the issue of the number of automata.

Moreover, this saves us the trouble of computing $P(B)$, a step whose complexity was $\mathcal{O}(n \cdot 8^{s'})$ in the procedure of [11]. Instead, since the definition of \mathcal{L}_π relies on $P(\pi)$, we only need to compute a description of $P(\pi)$ for $\pi \in B$. The detailed study of pin words done in Appendix B shows that this is possible from some quite simple tests on the decomposition tree of π , which will be detailed in Section 5. These tests are performed in $\mathcal{O}(\pi)$, so that the total cost of computing the description of $P(\pi)$ for all $\pi \in B$ is $\mathcal{O}(n)$.

To address the determinization issue, let us introduce a few notations. For any word $v = v_1 \dots v_p$ denote by $\overleftarrow{v} = v_p \dots v_1$ the reverse of v , and for any language \mathcal{L} , denote by $\overleftarrow{\mathcal{L}}$ the language $\{\overleftarrow{v} \mid v \in \mathcal{L}\}$. In practice, our algorithm does not test if $\mathcal{M} \setminus \bigcup_{\pi \in B} \mathcal{L}_\pi$ is finite, but rather whether its reverse language $\overleftarrow{\mathcal{M} \setminus \bigcup_{\pi \in B} \mathcal{L}_\pi}$ is finite. Notice that by definition of \mathcal{M} , we have $\overleftarrow{\mathcal{M} \setminus \bigcup_{\pi \in B} \mathcal{L}_\pi} = \overleftarrow{\mathcal{M}} \setminus \bigcup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi} = \mathcal{M} \setminus \bigcup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$. As in [11], to test the finiteness of this language, we will build an automaton $\mathcal{A}_\mathcal{C}$ accepting $\mathcal{M} \setminus \bigcup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$, and then test whether $\mathcal{A}_\mathcal{C}$ contains a cycle – see Subsection 5.3. Taking reverse languages is the trick that allows us to build a *deterministic* automaton $\mathcal{A}_\mathcal{C}$, thus avoiding the determinization which causes the complexity blow-up in the procedure of [11]. The connection between reverse and determinism is certainly unclear at the moment, but it is also hard to explain at this stage of the presentation of the algorithm. It follows from some details of Appendix B, and we will only explain this choice at the beginning of Appendix C.

We start with building, for each pin-permutation³ $\pi \in B$, a *deterministic* automaton \mathcal{A}_π which accepts $\overleftarrow{\mathcal{L}_\pi}$. An overview of the construction of these automata \mathcal{A}_π and of its complexity is given in the next subsection, while the details – that are quite technical – are postponed to Appendix C.

³Recall that when π is not a pin-permutation, both $P(\pi)$ and $\overleftarrow{\mathcal{L}_\pi}$ are empty.

From deterministic automata \mathcal{A}_π recognizing the languages $\overleftarrow{\mathcal{L}_\pi}$, we can obtain a deterministic automaton accepting words of $\cup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$ i.e., (up to intersection with \mathcal{M}) words whose reverses encode proper pin-permutations containing some pattern $\pi \in B$. To preserve determinism, the automaton accepting the union is not simply built by juxtaposition of the \mathcal{A}_π . Instead, we do the Cartesian product of the automata \mathcal{A}_π to compute a *deterministic* automaton accepting the union $\cup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$. This deterministic automaton can then be complemented in linear time, in order to build the automaton $\mathcal{A}_\mathcal{C}$ recognizing $\mathcal{M} \setminus \cup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$. Recall that the same operation on non-deterministic automata would be exponential in the worst case. The construction of $\mathcal{A}_\mathcal{C}$ will be detailed in Section 5, and its complexity analyzed. It is performed in $\mathcal{O}(s^{2k})$ where s and k are, as before, the maximal size of a pin-permutation in B and the number of pin-permutations in B . The complexity of testing with this method whether $\mathcal{C} = Av(B)$ contains finitely many proper pin-permutations is then also $\mathcal{O}(s^{2k})$. Writing that $\mathcal{O}(s^{2k}) = \mathcal{O}(2^{k \cdot 2 \log s})$ enables us to measure the complexity improvement w.r.t. the complexity $\mathcal{O}(2^{k \cdot s \cdot 2^s})$ of the procedure of [11]: the complexity gain is doubly exponential w.r.t. s .

4.3. Construction of the automata \mathcal{A}_π

The most difficult part of the algorithm outlined above is the construction of the deterministic automata \mathcal{A}_π accepting the languages $\overleftarrow{\mathcal{L}_\pi}$, for every pin-permutation $\pi \in B$. We give below the general idea of the method that is used for this construction, together with some results about the complexity of building \mathcal{A}_π . Detailed statements and proofs are provided in Appendices B and C.

From the definition of \mathcal{L}_π given p.14 we have:

$$\overleftarrow{\mathcal{L}_\pi} = \bigcup_{\substack{u \in P(\pi) \\ u = u^{(1)}u^{(2)} \dots u^{(j)}}} A^\star \overleftarrow{\phi(u^{(j)})} A^\star \dots A^\star \overleftarrow{\phi(u^{(2)})} A^\star \overleftarrow{\phi(u^{(1)})} A^\star$$

where $A = \{U, D, L, R\}$, ϕ is the map introduced in Definition 3.9 (p.12) and for every pin word u , by $u = u^{(1)}u^{(2)} \dots u^{(j)}$ we mean that $u^{(1)}u^{(2)} \dots u^{(j)}$ is the strong numeral-led factor decomposition of u . Therefore, we see that a description of $\overleftarrow{\mathcal{L}_\pi}$ will follow as soon as we are able to describe the set $P(\pi)$ of pin words of π , and more precisely their strong numeral-led factor decompositions.

In our previous work [6], we have given a recursive characterization of the decomposition trees of pin-permutations. We recall it below as Equation (\star) . We will then follow this characterization to recursively describe $P(\pi)$ for any pin-permutation π , and to subsequently give a recursive algorithm to build the automata \mathcal{A}_π recognizing $\overleftarrow{\mathcal{L}_\pi}$. We also present an alternative construction of \mathcal{A}_π whose complexity is optimized; but instead of $\overleftarrow{\mathcal{L}_\pi}$, the automaton recognizes a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$, which turns out to be sufficient for our purpose (see Subsection 5.3).

The recursive characterization of the decomposition trees of pin-permutations provided in [6] is as follows. It involves oscillations and quasi-oscillations, two special kinds of pin-permutations whose definitions are technical and given in Appendix A. The set \mathcal{S} of substitution decomposition trees of pin-permutations is recursively characterized by:

$$\begin{aligned} \mathcal{S} = & \bullet + \begin{array}{c} \oplus \\ \swarrow \quad \searrow \\ \mathcal{E}^+ \quad \mathcal{E}^+ \end{array} + \begin{array}{c} \oplus \\ \swarrow \quad \searrow \\ \mathcal{E}^+ \quad \mathcal{N}^+ \end{array} + \begin{array}{c} \ominus \\ \swarrow \quad \searrow \\ \mathcal{E}^- \quad \mathcal{E}^- \end{array} + \begin{array}{c} \ominus \\ \swarrow \quad \searrow \\ \mathcal{E}^- \quad \mathcal{N}^- \end{array} \\ & + \begin{array}{c} \alpha \\ \bullet \quad \bullet \end{array} + \begin{array}{c} \alpha \\ \bullet \quad \bullet \\ \vdots \\ \mathcal{S} \setminus \{\bullet\} \end{array} + \begin{array}{c} \beta^+ \\ \bullet \quad \bullet \\ \vdots \quad \vdots \\ \mathcal{S} \setminus \{\bullet\} \end{array} + \begin{array}{c} \beta^- \\ \bullet \quad \bullet \\ \vdots \quad \vdots \\ \mathcal{S} \setminus \{\bullet\} \end{array} \quad (*) \end{aligned}$$

where \mathcal{E}^+ (resp. \mathcal{E}^-) is the set of decomposition trees of increasing (resp. decreasing) oscillations, \mathcal{N}^+ (resp. \mathcal{N}^-) is the set of decomposition trees of pin-permutations that are not increasing (resp. decreasing) oscillations and whose root is not \oplus (resp. \ominus), α is any simple pin-permutation and β^+ (resp. β^-) is any increasing (resp. decreasing) quasi-oscillation. In Appendix A, in addition to recalling the definitions of increasing and decreasing (quasi-)oscillations, we present some of their properties. For the moment, let us only mention that some special pairs of points in quasi-oscillations can be identified, that are called *auxiliary* and *main substitution points*. Also, in every simple pin-permutation, we can identify special points, that are called *active points* whose definition will also be given in the appendix – see p.30. In Equation $(*)$ above, edges written $---$ (resp. $---$, $---$) correspond to an active point of α (resp. to a pair formed by an auxiliary point and a main substitution point of β^+ or β^-). In this equation the only terms that are recursive are those containing a subtree labeled by \mathcal{N}^+ , \mathcal{N}^- or $\mathcal{S} \setminus \{\bullet\}$.

Our characterization of $P(\pi)$ and construction of \mathcal{A}_π are divided into several cases, depending on which term of Equation $(*)$ π belongs to. First, we study the non-recursive cases, then the recursive cases with a linear root and finally the recursive cases with a prime root. Notice that the cases with root \ominus (resp. β^-) are up to symmetry⁴ identical to those with root \oplus (resp. β^+). We will therefore only consider the former in our analysis to follow.

Permutation of size 1. Notice first that the permutation $\pi = 1 = \bullet$ (whose decomposition tree is a leaf) has exactly four pin words – namely, $P(\pi) = \{1, 2, 3, 4\}$.

⁴This notion of symmetry is formalized by Remark B.1 in the appendix.

Then

$$\overleftarrow{\mathcal{L}}_\pi = \{A^* \overleftarrow{\phi(w)} A^* \mid w \in P(\pi)\} = A^* \mathcal{M}_2 A^*$$

where

$$\mathcal{M}_2 = \mathcal{M} \cap A^2 = \{UR, UL, DR, DL, RU, RD, LU, LD\}.$$

The language $\overleftarrow{\mathcal{L}}_\pi$ is recognized by the automaton \mathcal{A}_π of Figure 7.

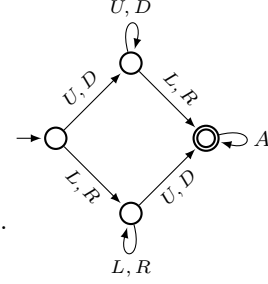


Figure 7: The automaton \mathcal{A}_π when $\pi = 1$.

Simple permutations. Let π be a simple pin-permutation. Theorem B.8 in Appendix B (p.38) shows that the number of pin words of π is at most 48. This of course does not describe the set $P(\pi)$ of pin words encoding π explicitly, but Algorithm 2 of [5] explains how to compute $P(\pi)$ in this case. We will get back to this computation in Section 5.

The construction of an automaton \mathcal{A}_π accepting $\overleftarrow{\mathcal{L}}_\pi$ is then explained in Remark C.6 in Appendix C. The time and space complexity of the construction of \mathcal{A}_π is quadratic w.r.t. $|\pi|$, as soon as the pin words of π are given. In Remark C.8 in Appendix C, we explain how to improve the complexity of the construction of \mathcal{A}_π , so that it is linear in time and space. The automaton \mathcal{A}_π so obtained however does not accept $\overleftarrow{\mathcal{L}}_\pi$ but a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$.

Non recursive case with a linear root. W.l.o.g., we consider $\pi = \oplus[\xi_1, \dots, \xi_r]$ where ξ_i are increasing oscillations. The set of pin words of π is expressed using the shuffle product \sqcup (defined p.39) in Theorem B.12 of Appendix B. Namely, denoting $P^{(1)}(\xi_k)$ (resp. $P^{(3)}(\xi_k)$) the set of pin words that encode ξ_k and whose origin lies in quadrant 1 (resp. 3) with respect to the points of ξ_k , we have:

$$P(\pi) = \bigcup_{1 \leq i \leq r-1} P(\oplus[\xi_i, \xi_{i+1}]) \cdot \left((P^{(1)}(\xi_{i-1}), \dots, P^{(1)}(\xi_1)) \sqcup (P^{(3)}(\xi_{i+2}), \dots, P^{(3)}(\xi_r)) \right).$$

Together with Lemmas B.13 and B.18, this provides an explicit expression of $P(\pi)$.

The automaton \mathcal{A}_π accepting $\overleftarrow{\mathcal{L}}_\pi$ (resp. a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$) is built by assembling smaller automata, which correspond to the different languages that appear in the shuffle product expression of $P(\pi)$. Subsection C.3 gives the details of this construction, and Theorem C.16 (resp. Remark C.17) proves its correctness. Lemma C.12 shows that it is achieved in time and space $\mathcal{O}(|\pi|^4)$ (resp. $\mathcal{O}(|\pi|^2)$). The automaton \mathcal{A}_π in this case is shown in Figure 23 (p.61).

Recursive case with a linear root. W.l.o.g., we consider a permutation $\pi = \oplus[\xi_1, \dots, \xi_\ell, \rho, \xi_{\ell+2}, \dots, \xi_r]$, where all ξ_i are increasing oscillations, but ρ is

not. By induction, we may assume that we have an explicit description of $P(\rho)$, and an automaton \mathcal{A}_ρ which accepts $\overleftarrow{\mathcal{L}}_\rho$ (resp. a language \mathcal{L}'_ρ such that $\mathcal{L}'_\rho \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\rho \cap \mathcal{M}$). In Appendices B and C, the decomposition tree of ρ is denoted T_{i_0} .

The set $P(\pi)$ of pin words of π (which of course depends on $P(\rho)$) is given by Theorem B.19 in Appendix B. It always contains

$$P_0 = P(\rho) \cdot (P^{(1)}(\xi_\ell), \dots, P^{(1)}(\xi_1)) \sqcup (P^{(3)}(\xi_{\ell+2}), \dots, P^{(3)}(\xi_r))$$

but it may contain more words, if π satisfies additional conditions that are shown in the middle two columns of Figure 17 (p.44). In all these possible cases (considered up to symmetry), Theorem B.19 describes explicitly the complete set of pin words of π .

If $P(\pi) = P_0$, like in the previous case the automaton \mathcal{A}_π accepting $\overleftarrow{\mathcal{L}}_\pi$ is built by assembling \mathcal{A}_ρ with small automata corresponding to the different languages in the shuffle product defining P_0 . The automaton \mathcal{A}_π so obtained is shown in Figure 24 (p.65).

Otherwise, $P_0 \subsetneq P(\pi)$, so that the automaton of Figure 24 accepts some but not all words of $\overleftarrow{\mathcal{L}}_\pi$. It is however possible to modify it by adding some transitions, so that the resulting automaton \mathcal{A}_π accepts exactly $\overleftarrow{\mathcal{L}}_\pi$. These modifications of the automaton are shown in the last column of Figure 17 (p.44).

These constructions are explained in Subsection C.4. The proof that they are correct are however omitted even in the appendix, as they are very similar to some other proofs that are detailed there. The complexity of these constructions are given in Lemmas C.19 and C.21: it is done in time and space $\mathcal{O}((|\pi| - |\rho|)^2)$ plus the additional complexity due to the construction of \mathcal{A}_ρ . The construction and its complexity are not modified (except for the recursive part) for building an automaton \mathcal{A}_π accepting a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$ instead of $\overleftarrow{\mathcal{L}}_\pi$.

Recursive case with a prime root. We start with the case where $\pi = \alpha[1, \dots, 1, \rho, 1, \dots, 1]$ with α a simple pin-permutation and $\rho \neq 1$. The only other possibility is that π is a permutation whose decomposition tree has a root labeled by a quasi-oscillation with two children that are not leaves. This special case will be considered in the next paragraph.

By induction, we may assume that we have an explicit description of $P(\rho)$, and an automaton \mathcal{A}_ρ which accepts $\overleftarrow{\mathcal{L}}_\rho$ (resp. a language \mathcal{L}'_ρ such that $\mathcal{L}'_\rho \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\rho \cap \mathcal{M}$). We denote by x the point of α expanded by ρ . Let us also record here that the decomposition tree of ρ will be denoted T in Appendices B and C.

As in Definition B.23 (p.48), let $Q_x(\alpha)$ denote the set of strict pin words obtained by deleting the first letter of a quasi-strict pin word encoding a pin representation of α starting in x . The set $P(\pi)$ of pin words of π is given by Theorem B.25 (p.49) in Appendix B. As in the previous case, it always contains a ground set of words, in this case $P(\rho) \cdot Q_x(\alpha)$, but it contains more

words in case π satisfies some further condition – denoted (\mathcal{C}) in Appendices B and C. Although it becomes quite technical, it is possible to make explicit the description of $P(\pi)$ in Theorem B.25, using Lemma B.27 and Remarks B.26, B.29 and B.30.

When $P(\pi) = P(\rho) \cdot Q_x(\alpha)$, the automaton \mathcal{A}_π which accepts $\overleftarrow{\mathcal{L}}_\pi$ is easily constructed from \mathcal{A}_ρ and $Q_x(\alpha)$. As before, when $P(\rho) \cdot Q_x(\alpha) \subsetneq P(\pi)$, \mathcal{A}_π has the same general structure as this automaton, with some new transitions added. This is shown on Figure 25 (p.69). Theorem C.23 proves the correctness of this construction. This however only holds for $|\rho| \neq 2$. In the special case $|\rho| = 2$, the construction of \mathcal{A}_π is not recursive anymore, and is easily solved (see p.70).

The complexity of the construction of \mathcal{A}_π is discussed in Lemma C.25. Except for the special case $|\rho| = 2$, it is $\mathcal{O}(|\pi| - |\rho|)$ in time and space, in addition to the complexity of computing \mathcal{A}_ρ . This holds both for the automaton \mathcal{A}_π accepting $\overleftarrow{\mathcal{L}}_\pi$ and for its variant (whose construction is unchanged except for the recursive part) which accepts a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$. For the special case $|\rho| = 2$, the complexity of building \mathcal{A}_π accepting $\overleftarrow{\mathcal{L}}_\pi$ is $\mathcal{O}(|\pi|^3)$, and it drops to $\mathcal{O}(|\pi|^2)$ for the variant accepting \mathcal{L}'_π .

The special case of increasing quasi-oscillations. W.l.o.g., the only remaining case in Equation (\star) is that of a permutation $\pi = \beta^+[1, \dots, 1, \rho, 1, \dots, 1, 12, 1, \dots, 1]$ where β^+ is an increasing quasi-oscillation, the permutation 12 expands an auxiliary point of β^+ and ρ (of size at least 2) expands the corresponding main substitution point of β^+ . Again, in Appendices B and C, the decomposition tree of ρ is denoted T . And by induction, we may assume that we have an explicit description of $P(\rho)$, and an automaton \mathcal{A}_ρ which accepts $\overleftarrow{\mathcal{L}}_\rho$ (resp. a language \mathcal{L}'_ρ such that $\mathcal{L}'_\rho \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\rho \cap \mathcal{M}$).

This case is very much constrained, and the set of pin words of π is completely determined by Theorem B.31(p.52): $P(\pi) = P(\rho) \cdot w$, for some word w which is uniquely determined, and that Remark B.32 shows explicitly. From this, it is not hard to build from \mathcal{A}_ρ an automaton \mathcal{A}_π which accepts $\overleftarrow{\mathcal{L}}_\pi$ (resp. a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$). This is explained in Paragraph C.5.2 (p.71), and is performed in $\mathcal{O}(|\pi| - |\rho|)$ time and space in addition to the time and space complexity of the construction of \mathcal{A}_ρ .

The automaton \mathcal{A}_π associated with a pin-permutation π is then build recursively, by first determining which shape of tree in Equation (\star) is matched by the decomposition tree of π , and then applying the corresponding construction. From the complexities of these constructions, it is not hard to evaluate the overall complexity of building \mathcal{A}_π . Namely:

Theorem 4.1. *Given π a pin-permutation, the above recursive construction allows to build an automaton \mathcal{A}_π which accepts $\overleftarrow{\mathcal{L}}_\pi$ (resp. a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$) in time and space complexity $\mathcal{O}(|\pi|^4)$ (resp. $\mathcal{O}(|\pi|^2)$).*

The proof of Theorem 4.1 is however postponed to the appendix – see Theorem C.26 (p.74). Indeed, to prove this theorem, we need to be careful about some

details of the construction of automata, that are only explained in the appendix. The main difficulty is to ensure that some special states in the automata can be “marked” in some of the above constructions without increasing the complexity. The marking of these special states is needed to build the additional transitions in the cases $\pi = \oplus[\xi_1, \dots, \xi_\ell, \rho, \xi_{\ell+2}, \dots, \xi_r]$ and $\pi = \alpha[1, \dots, 1, \rho, 1, \dots, 1]$, since these transitions are actually pointing towards these “marked” states. Subsection C.6 explains how to mark these special states along the construction of \mathcal{A}_π .

5. A polynomial algorithm deciding whether a class contains a finite number of simple permutations

In this section, we prove the main result of this article:

Theorem 5.1. *Given a finite set of permutations B , we describe an algorithm that determines whether the permutation class $\mathcal{C} = \text{Av}(B)$ contains a finite number of simple permutations. Denoting $n = \sum_{\pi \in B} |\pi|$, $p = \prod |\pi|$ where the product is taken over all pin-permutations in B , k the number of pin-permutations in B and s the maximal size of a pin-permutation of B , the complexity of the algorithm is $\mathcal{O}(n \log n + s^{2k})$ or more precisely $\mathcal{O}(n \log n + p^2)$.*

The complexity which is achieved in Theorem 5.1 makes use of the optimized variant of the construction of the automata \mathcal{A}_π . Notice that with the non-optimized construction of the automata \mathcal{A}_π , although we would have an algorithm whose details are a little bit simpler to describe, its complexity would be significantly worst than with the optimized variant, namely $\mathcal{O}(n \log n + p^4) = \mathcal{O}(n \log n + s^{4k})$.

The algorithm announced in Theorem 5.1 can be decomposed into several steps and is described in the rest of this section.

5.1. Finitely many parallel alternations and wedge simple permutations in \mathcal{C} ?

Following [11] (see Theorem 2.8 p.7) we first check whether \mathcal{C} contains finitely many parallel alternations and wedge simple permutations. From Lemmas 2.9, 2.10 and 2.11 (p.8) 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 (see Lemma 2.12 p.8).

5.2. Finding pin-permutations in the basis

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 [6], and recalled in Equation (★) (p.20).

More precisely, for each $\pi \in B$, we proceed as follows.

- First we compute its decomposition tree T_π .

This is achieved in linear time w.r.t. $|\pi|$, computing first the skeleton of T_π following [7] or [12], and next the labels of linear and prime nodes as explained in [8, §2.2].

- Second we add some information on the decomposition tree.

This information will be useful in later steps of our algorithm to check whether π is a pin-permutation, and next (in the affirmative) to determine which construction of the automaton \mathcal{A}_π (see Subsection 4.3 or details in Appendix C) applies to π .

- For each prime node N , we record whether the simple permutation α labeling N is an increasing or decreasing oscillation or quasi-oscillation.

This may be recorded by performing a linear time depth-first traversal of T_π , and checking each node when it is reached. As there are 4 oscillations of each size that are explicitly described as 2416385... (see Figure 9 p.31) or one of its symmetries, checking if a simple permutation α is of this form can be done in linear time w.r.t. $|\alpha|$. The same kind of explicit description also holds for quasi-oscillations, and in addition we can record which children correspond to the auxiliary and main substitution points.

- For each node N , we record whether the subtree rooted at N encodes an increasing or decreasing oscillation.

This may be recorded easily, along the same depth-first traversal of T_π as above. Indeed oscillations of size greater than 3 are simple permutations, and increasing (resp. decreasing) oscillations of smaller sizes are 1, 21, 231 and 312 (resp. 1, 12, 132 and 213). So it is sufficient to check whether N is a leaf, or a prime node labeled by an increasing (resp. decreasing) oscillation all of whose children are leaves, or a linear node with exactly two children satisfying extra constraints: they are either both leaves, or one is a leaf and the second one is a linear node with exactly two children that are both leaves. In this later case the oscillation is increasing (resp. decreasing) if N is labeled \ominus (resp. \oplus).

These computations are performed in linear time w.r.t. $|\alpha|$ for any prime node labeled by $|\alpha|$, and in constant time for any linear node. Hence, as the sum of the sizes of the labels of all internal nodes is linear w.r.t. $|\pi|$, the overall complexity of this step is linear w.r.t. $|\pi|$.

- Finally we determine whether π is a pin-permutation or not.

To do so, we recursively check starting with the root whether its decomposition tree is of the shape described in [6] (see Equation (\star) p.20).

- If the root is linear, with the additional information stored we can check whether all its children are increasing (resp. decreasing) oscillations in linear time w.r.t. the number of children. If exactly one child is not an increasing (resp. decreasing) oscillation, we check recursively whether the subtree rooted at this child is the decomposition tree of a pin-permutation.

- If the root is prime, we first check whether its label α is a *pin*-permutation. More precisely, with Algorithm 2 of [5] we compute the set of pin words of α and test its emptiness. By Lemma 4.1 of [5], this is done in linear time w.r.t. $|\alpha|$. Then we check whether all the children of the root are leaves.

- If exactly one child is not a leaf, we furthermore have to check whether the point x it expands is an active point of α . With some precisions given in the appendix, this can be done in $\mathcal{O}(|\alpha|)$ time. Namely, from Remark B.24

(p.49) we just have to test the emptiness of $Q_x(\alpha)$, which is computed in linear time w.r.t. $|\alpha|$ (see Remark B.26 p.50). Then we check recursively whether the subtree rooted at x is the decomposition tree of a pin-permutation.

- If exactly two children are not leaves, with the additional information stored we can check in constant time whether α is an increasing (resp. decreasing) quasi-oscillation, if the two children that are not leaves expand the auxiliary and main substitution points, and if the one expanding the auxiliary point is the permutation 12 (resp. 21). Then we check recursively whether the subtree rooted at the main substitution point is the decomposition tree of a pin-permutation.

As the complexity of each step is linear w.r.t. the number of children (which is also the size of the label for a prime node), deciding whether a permutation π is a pin-permutation or not can be done in linear time w.r.t. $|\pi|$. The overall determination of PB is therefore linear in $n = \sum_{\pi \in B} |\pi|$.

Moreover, in addition to computing PB , the above procedure produces additional results, that we also record as they are useful in the next step. Namely, for every permutation π of PB , we record its decomposition tree T_π , together with the additional information computed on its nodes; and we also record the set of pin words that encode each simple permutation α labeling a prime node N of T_π and the set $Q_x(\alpha)$ when N has exactly one non-trivial child. Notice that the knowledge of these is sufficient to characterize the set of pin words that encode π thanks to results of Appendix B outlined in Subsection 4.3.

5.3. Finitely many proper pin-permutations in \mathcal{C} ?

From Theorem 3.18 (p.15) it is enough to check whether $\mathcal{M} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$ is finite. This can be easily decided with a deterministic automaton $\mathcal{A}_\mathcal{C}$ recognizing $\overleftarrow{\mathcal{M} \setminus \cup_{\pi \in B} \mathcal{L}_\pi}$. From the previous step of the procedure, we know the set PB of pin-permutations of B and some additional results described above. First notice that $\cup_{\pi \in B} \mathcal{L}_\pi = \cup_{\pi \in PB} \mathcal{L}_\pi$ as \mathcal{L}_π is empty when π is not a pin-permutation (see p.14). We build the automaton $\mathcal{A}_\mathcal{C}$ as follows.

- First for each pin-permutation $\pi \in PB$, we construct \mathcal{A}_π – which is deterministic and complete – recognizing a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$. For this optimized variant, the construction is performed in time and space at most $\mathcal{O}(|\pi|^2)$ as presented in Subsection 4.3 and described in details in Appendix C (see Theorem 4.1 p.23 or Theorem C.26 p.74). Notice that the construction of \mathcal{A}_π depends on the shape of the decomposition tree T_π of π . But thanks to the additional information stored in T_π , we can determine which tree shape matches T_π in linear time w.r.t. the number of children of the root of T_π , and the same holds at each recursive step of the construction.

- Then we build a deterministic automaton \mathcal{A}_1 recognizing $\bigcup_{\pi \in PB} \mathcal{L}'_\pi$, where \mathcal{L}'_π is defined as in the first item. The automaton \mathcal{A}_1 is obtained performing the deterministic union (as a Cartesian product, see [16] for details) of all the automata \mathcal{A}_π . This is done in time and space $\mathcal{O}(\prod_{\pi \in PB} |\mathcal{A}_\pi|) = \mathcal{O}(\prod_{\pi \in PB} |\pi|^2)$.

• Then we build the automaton \mathcal{A}_2 which is the deterministic intersection (again as a Cartesian product) between \mathcal{A}_1 and the automaton $\mathcal{A}(\mathcal{M})$ given in Figure 8 in time and space $\mathcal{O}(|\mathcal{A}_1| \cdot |\mathcal{A}(\mathcal{M})|) = \mathcal{O}(\prod_{\pi \in PB} |\pi|^2)$.

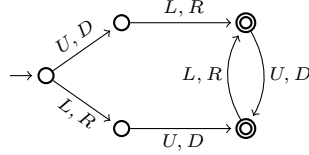


Figure 8: A deterministic automaton $\mathcal{A}(\mathcal{M})$ recognizing the set \mathcal{M} of words of length at least 2 without any factor in $\{UU, UD, DU, DD, RR, RL, LR, LL\}$.

The automaton \mathcal{A}_2 recognizes $(\bigcup_{\pi \in PB} \mathcal{L}'_{\pi}) \cap \mathcal{M} = (\bigcup_{\pi \in PB} \overleftarrow{\mathcal{L}}_{\pi}) \cap \mathcal{M}$. By Lemma 3.17 (p.15) this is the language of words $\overleftarrow{\phi(w)}$ for all strict pin words w encoding permutations having a pattern in PB , *i.e.* that are not in \mathcal{C} . Notice that by Remark 3.5 (p.11) such permutations are necessarily proper pin-permutations.

• Next we complement \mathcal{A}_2 to build a deterministic automaton \mathcal{A}_3 recognizing $A^* \setminus ((\bigcup_{\pi \in PB} \overleftarrow{\mathcal{L}}_{\pi}) \cap \mathcal{M})$. As \mathcal{A}_2 is deterministic, its complement is obtained in linear time w.r.t. its size, by completing it and then turning every final (resp. non-final) state into a non-final (resp. final) state. Moreover the size of \mathcal{A}_3 is the same as that of the automaton obtained completing \mathcal{A}_2 , *i.e.*, $\mathcal{O}(\prod_{\pi \in PB} |\pi|^2)$

• Finally we compute the deterministic intersection between \mathcal{A}_3 and the automaton $\mathcal{A}(\mathcal{M})$ to obtain the automaton $\mathcal{A}_{\mathcal{C}}$. This is done in time and space $\mathcal{O}(|\mathcal{A}_3| \cdot |\mathcal{A}(\mathcal{M})|) = \mathcal{O}(\prod_{\pi \in PB} |\pi|^2)$. The automaton $\mathcal{A}_{\mathcal{C}}$ built in this way recognizes $\mathcal{M} \setminus (\bigcup_{\pi \in PB} \overleftarrow{\mathcal{L}}_{\pi}) = \mathcal{M} \setminus (\bigcup_{\pi \in \mathcal{C}} \mathcal{L}_{\pi})$. This is the language of all words $\overleftarrow{\phi(w)}$ where w is a strict pin word encoding a permutation of \mathcal{C} (that is necessarily a proper pin-permutation, as above).

Then, by Theorem 3.18 (p.15), checking whether the permutation class \mathcal{C} contains a finite number of proper pin-permutations is equivalent to checking whether the language recognized by $\mathcal{A}_{\mathcal{C}}$ is finite *i.e.*, whether $\mathcal{A}_{\mathcal{C}}$ does not contain any cycle that is accessible and co-accessible (*i.e.*, a cycle that can be reached from an initial state and from which a final state can be reached). The automaton $\mathcal{A}_{\mathcal{C}}$ is not necessarily accessible and co-accessible. Its accessible part is made of all states that can be reached in a traversal of the automaton from the initial state; its co-accessible part is obtained similarly by a traversal from the set of final states taking the edges of the automaton backwards. Before looking for a cycle, we make $\mathcal{A}_{\mathcal{C}}$ accessible and co-accessible by keeping only its accessible and co-accessible part, yielding a smaller automaton $\mathcal{A}'_{\mathcal{C}}$. The complexity of this double reduction of the size of the automaton is linear in time

w.r.t. the size of $\mathcal{A}_{\mathcal{C}}$. Moreover the size of $\mathcal{A}'_{\mathcal{C}}$ is smaller than or equal to the one of $\mathcal{A}_{\mathcal{C}}$, *i.e.*, $\mathcal{O}(\prod_{\pi \in PB} |\pi|^2)$. Finally we test whether $\mathcal{A}'_{\mathcal{C}}$ does not contain any cycle. This can be done in $\mathcal{O}(|\mathcal{A}'_{\mathcal{C}}|)$ time with a depth-first traversal of $\mathcal{A}'_{\mathcal{C}}$.

Let s be the maximal size of a pin-permutation of B and k the number of pin-permutations in B , then $\mathcal{O}(\prod_{\pi \in PB} |\pi|^2) = \mathcal{O}(s^{2k})$. Hence putting all these steps together leads to an algorithm whose complexity is $\mathcal{O}(s^{2k})$ to check whether there are finitely many proper pin-permutations in \mathcal{C} , when the set PB of pin-permutations of B , their decomposition trees and the set of pin words of each simple permutation appearing in these trees are given.

6. Conclusion

The work reported here follows the line opened by [3] and continued by [11]. 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, [11] 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 [6], we have described in the above an algorithm testing this condition. The analysis of its complexity shows that it is efficient.

Because an algebraic generating function is a witness of the combinatorial structure of a permutation class, we may interpret our result as giving an efficient algorithm testing a sufficient condition for a permutation class to be well-structured. We believe that more could and should be done on the algorithmization of finding structure in permutation classes. In particular, we plan to provide efficient algorithms that do not only *test* that there is an underlying structure in a permutation class, but that also *compute* this structure. We set in the sequel the main steps towards the achievement of this project.

As discussed in [3], the proof of the main theorem therein is constructive. Namely, given the basis B of a class \mathcal{C} , and the set $\mathcal{S}_{\mathcal{C}}$ of simple permutations in \mathcal{C} (assuming that both are finite), the proof of the main theorem of [3] describes how to compute (a polynomial system satisfied by) the generating function of \mathcal{C} , proving thereby that it is algebraic. The main step is actually to compute a (possibly ambiguous) context-free grammar of trees for the permutations of \mathcal{C} , or rather their decomposition trees.

Such a context-free grammar of trees almost captures the combinatorial structure of a permutation class. The only reason why it does not completely is because the grammar may be ambiguous, and thus may generate several times the same permutation in the class. On the contrary, unambiguous context-free grammars of trees fall exactly in the context of the *combinatorial specifications* of [13], and describing a permutation class by such a combinatorial specification is undoubtedly demonstrating the structure of the class. Consequently, we aim

at describing an algorithm to compute this combinatorial specification, assuming we are given the finite basis B characterizing the class \mathcal{C} . There would be four main steps in such an algorithm.

First, we should ensure that \mathcal{C} falls into the set of permutation classes we can handle, *i.e.*, ensure that \mathcal{C} contains a finite number of simple permutations. The present work gives an algorithm for this first step.

Second, when finite, we should compute the set $\mathcal{S}_{\mathcal{C}}$ of simple permutations in \mathcal{C} . A naive method to do so can be immediately deduced from the results of [3], but it is of highly exponential complexity. An algorithm for this second step has subsequently been described in [18], and its complexity analyzed. It should be noticed that the complexity of this algorithm also depends on the size of its output, namely on $|\mathcal{S}_{\mathcal{C}}|$ and on $\max\{|\pi| : \pi \in \mathcal{S}_{\mathcal{C}}\}$.

Third, from B and $\mathcal{S}_{\mathcal{C}}$, we should turn the constructive proof of [3] into an actual algorithm, that would compute the (possibly ambiguous) context-free grammar of trees describing the decomposition trees of the permutations of \mathcal{C} .

Finally, we should transform this (possibly ambiguous) context-free grammar into an unambiguous combinatorial specification for \mathcal{C} . We have described in the extended abstract [4] an algorithm for these last two steps, whose complexity is still to analyze.

Combining these four steps will provide an algorithm to obtain from a basis B of excluded patterns a combinatorial specification for the permutation class $\mathcal{C} = Av(B)$. We are not only convinced of the importance of this result from a theoretical point of view, but also (and maybe more importantly) we are confident that it will be of practical use to the permutation patterns community. Indeed, from a combinatorial specification, it is of course possible with the methodology of [13] to immediately deduce a system of equations for the generating function of \mathcal{C} . But other algorithmic developments can be considered. In particular, this opens the way to obtaining systematically uniform random samplers of permutations in a class, or to the automatic evaluation of the Stanley-Wilf growth rate of a class.

Acknowledgments. We are very grateful to the anonymous referee for providing both specific comments and global suggestions on the organization of our paper. These helped us improve the presentation of our work, on both the large and small scales. We would also like to thank Joseph Kung for his availability and efficiency as an editor.

Appendices

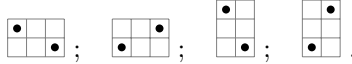
A. Simple pin-permutations, oscillations and quasi-oscillations

This appendix groups together some technical definitions and results about subsets of pin-permutations: the simple ones, the oscillations, and the quasi-oscillations. The last two play an important role in the characterization of

substitution decomposition trees associated with pin-permutations (see Equation (\star) p.20).

A.1. Simple pin-permutations, active knights and active points

Let σ be a simple pin-permutation. We have seen from Remark 2.7 (p.7) that all pin representations of σ are proper. This implies in particular (see [6, Lemma 4.3] for an immediate proof) that the first two points in every pin representation of σ are in *knight position*, i.e., form one of the following configurations in the diagram of σ :



We define an *active knight* of σ to be a pair of points of σ in knight position which is the possible start of a pin representation of σ . The definition of active knights may be extended to pin-permutations σ that are not necessarily simple, as pairs of points of σ that are the possible start of a pin representation of σ . In addition to the four configurations shown above, such active “knights” of non-simple pin-permutations may form a configuration $\begin{array}{|c|c|} \hline \bullet & \bullet \\ \hline & \\ \hline \end{array}$ or $\begin{array}{|c|c|} \hline & \bullet \\ \hline \bullet & \bullet \\ \hline \end{array}$ in the diagram of σ . We also define an *active point* of σ to be a point of σ belonging to an active knight of σ . The active knights (and hence the active points) of any simple pin-permutation may be described (see [6, Lemma 4.6]). This is however not needed for us in this work, except in the case of oscillations, which we will review in the following.

A.2. Oscillations

Following [11], let us consider the infinite oscillating sequence defined by $\omega = 3\ 1\ 5\ 2\ 7\ 4\ 9\ 6\ \dots (2k+1)\ (2k-2)\ \dots$. The leftmost part of Figure 9 shows the diagram of a prefix of ω .

Definition A.1. An increasing oscillation of size $n \geq 4$ is a simple permutation of size n that is contained as a pattern in ω . For smaller sizes the increasing oscillations are 1, 21, 231 and 312. A decreasing oscillation is the reverse⁵ of an increasing oscillation.

There are two increasing oscillations of any size greater than or equal to 3, that can also be given explicitly. For even size, they are

$$2\ 4\ 1\ 6\ 3\ \dots (2k+2)\ (2k-1)\ \dots (2n)\ (2n-3)\ (2n-1)\ \text{and} \\ 3\ 1\ 5\ 2\ 7\ 4\ \dots (2k+1)\ (2k-2)\ \dots (2n)\ (2n-2);$$

for odd size,

$$2\ 4\ 1\ 6\ 3\ \dots (2k+2)\ (2k-1)\ \dots (2n)\ (2n-3)\ (2n+1)\ (2n-1)\ \text{and} \\ 3\ 1\ 5\ 2\ 7\ 4\ \dots (2k+1)\ (2k-2)\ \dots (2n+1)\ (2n-2)\ (2n).$$

⁵The reverse of $\sigma = \sigma_1\sigma_2\ \dots\ \sigma_n$ is $\overleftarrow{\sigma} = \sigma_n\ \dots\ \sigma_2\sigma_1$.

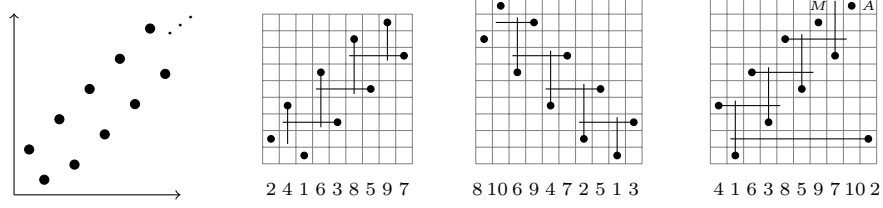


Figure 9: The infinite oscillating sequence ω , an increasing oscillation ξ of size 9, a decreasing oscillation of size 10, and an increasing quasi-oscillation of size 10 (obtained from ξ by addition of a maximal element or equivalently by taking the inverse of the explicit quasi-oscillation of size 10 given in Subsection A.3), with a pin representation for each.

A similar statement holds for decreasing oscillations. As noticed in [6, Lemma 2.23], every increasing (resp. decreasing) oscillation is a pin-permutation. Moreover, by definition all oscillations of size at least 4 are simple. Finally, notice also that permutations 1, 2 4 1 3 and 3 1 4 2 are both increasing and decreasing oscillations, and are the only ones with this property. The middle two diagrams of Figure 9 show some examples of oscillations.

In Appendix B, we will describe explicitly the set of pin words of all pin-permutations. In Subsection B.2, this requires some knowledge about the pin words of oscillations (w.l.o.g., only increasing oscillations) – see in particular Lemmas B.13 to B.18. In these lemmas, we have to distinguish cases according to the active knights of the increasing oscillations. For this reason, we review in the sequel some results from [6] about the active knights of increasing oscillations.

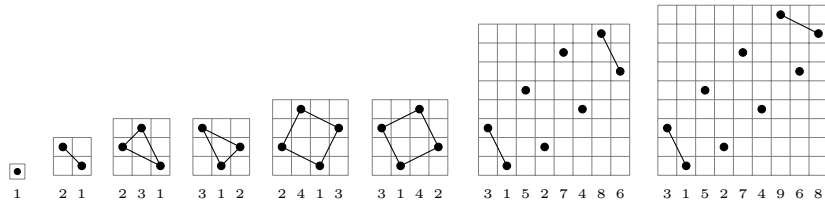


Figure 10: The increasing oscillations of size less than 5 and two increasing oscillations respectively of size 8 with type (V, V) and 9 with type (V, H) . Active knights are marked by edges between their two active points.

Figure 10 shows the active knights of the increasing oscillations of size up to 4. Lemma 4.6 of [6] describes the active knights of simple pin-permutations, and in particular those of the increasing oscillations of size at least 4. It follows from this lemma that an increasing oscillation of size at least 5 has exactly two active

knights. They are located at both ends of the main diagonal and they consist of two points in relative order 21 (see Figure 10). These active knights are either in horizontal (H) position $\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}$ or in vertical (V) position $\begin{smallmatrix} \blacksquare & \\ \blacksquare & \blacksquare \end{smallmatrix}$. Therefore there are four types of increasing oscillations of size at least 5: (x, y) with $x, y \in \{H, V\}$, where x is the type of the lower left active knight and y for the upper right. This definition can be extended to increasing oscillations of size 4, considering their two active knights in relative order 21 (see Figure 10). Note that an even size oscillation has type (H, H) or (V, V) and an odd size one (H, V) or (V, H) .

A.3. Quasi-oscillations

We recall the definition of quasi-oscillations from [6].

Definition A.2. 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 the move of an element of ξ according to the rules of Table 1⁶.

Element inserted	Pattern $\xi_1 \xi_2 \xi_3$	Pattern $\xi_{n-3} \xi_{n-2} \xi_{n-1}$	Element to move 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: Building quasi-oscillations from oscillations, and defining their main substitution points.

We define the auxiliary point (A) to be the point added to ξ , and the main substitution point (M) to be an extremal point of ξ according to Table 1.

Furthermore, for $n = 4$ or 5 , there are two increasing quasi-oscillations of size n : $2413, 3142, 25314$ and 41352 . Each of them has two possible choices for its pair of auxiliary and main substitution points. See Figure 11 for more details. Finally, a decreasing quasi-oscillation is the reverse of an increasing quasi-oscillation.

In particular, it follows from Definition A.2 that the auxiliary point of increasing quasi-oscillations of size at least 6 is uniquely determined, whereas

⁶ The first row of Table 1 reads as follows: If a maximal element is added to ξ , with $\xi \in S_{n-1}$ starting (resp. ending) with a pattern 231 (resp. 132), then the corresponding increasing quasi-oscillation β is obtained by moving the left-most point of ξ so that it becomes the right-most (in β), and the main substitution point is the largest point of ξ (see the rightmost diagram of Figure 9).

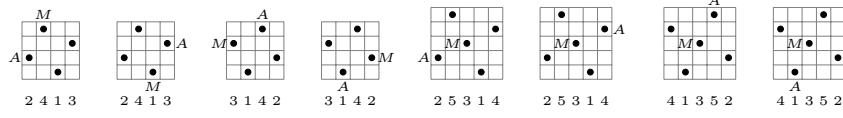


Figure 11: The diagrams of the increasing quasi-oscillations of size 4 and 5, where auxiliary (A) and main (M) substitution points are marked.

there are two possible choices of auxiliary point in increasing quasi-oscillations of size 4 and 5.

As noticed in [6] there are four increasing (resp. decreasing) quasi-oscillations of size n for any $n \geq 6$, two of size 4 (2 4 1 3 and 3 1 4 2) and two of size 5 (2 5 3 1 4 and 4 1 3 5 2). Moreover, every quasi-oscillation is a simple pin-permutation.

Like oscillations, the quasi-oscillations of size at least 6 may also be defined explicitly. Namely, one increasing quasi-oscillation is

$$4\ 1\ 6\ 3\ \dots\ (2k+2)\ (2k-1)\ \dots\ (2n-2)\ (2n-5)\ \underbrace{(2n-1)\ (2n-3)}_M\ \underbrace{(2n)}_A\ 2\ \text{for even size}$$

$$4\ 1\ 6\ 3\ \dots\ (2k+2)\ (2k-1)\ \dots\ (2n-5)\ (2n)\ \underbrace{(2n-3)\ (2n-1)}_M\ \underbrace{(2n+1)}_A\ 2\ \text{for odd size,}$$

where M (resp. A) indicates the main substitution point (resp. the auxiliary point). The other three increasing quasi-oscillations are obtained applying some symmetries to the diagram of the above permutation σ , namely reflexion according to its two diagonals. In other words, the four increasing quasi-oscillations are σ , its so-called *reverse-complement* σ^{rc} , and their inverses σ^{-1} and $(\sigma^{rc})^{-1}$. The definition of the auxiliary and main substitution points follows along the application of these symmetries.

It should be noticed that each quasi-oscillation of size 4 or 5 is both increasing and decreasing. However, once its auxiliary point is chosen among the four possibilities, then its nature (increasing or decreasing) is determined without ambiguity, and so is its main substitution point. Moreover, knowing the (unordered) pair of points which are the auxiliary and main substitution points, we can deduce which one is the auxiliary point without ambiguity.

We conclude this paragraph about quasi-oscillations with a remark on the number of their active knights which involve (one of) their auxiliary point(s). This information will be useful in the proof of Lemma B.27 (p.50).

Remark A.3. *The auxiliary point of an increasing quasi-oscillation of size n (or any of its auxiliary points, in case $n = 4$ or 5) belongs to exactly one active knight if $n \neq 4$, and to exactly two active knights if $n = 4$.*

Proof. Consider first increasing quasi-oscillations of size greater than 5. From Lemma 4.6 of [6] (see also the last diagram of Figure 9), the main substitution point belongs to exactly two active knights – one formed with the auxiliary

point and one formed with the point separating it from the auxiliary point – and there are no other active knights.

Consider now an increasing quasi-oscillation of size 4 or 5 (see Figure 11) where an auxiliary point x is chosen. We may also apply to Lemma 4.6 of [6] to count its active knights that involve x . Namely, an increasing quasi-oscillation of size 5 has exactly 4 active knights, all of them contain the main substitution point (which is uniquely determined, regardless of the choice of x), and exactly one of them contains the auxiliary point x . Finally, an increasing quasi-oscillation of size 4 has exactly 4 active knights and each of its points (including the auxiliary point x) belongs to exactly two active knights. \square

We refer the reader to [6] for further properties of oscillations and quasi-oscillations.

B. Pin words of pin-permutations

Our goal here is to describe the set $P(\pi)$ of pin words that encode a pin-permutation π , following the recursive characterization of the decomposition trees of pin-permutations that is given by Equation (\star) (p.20).

As outlined in Subsection 4.3, the characterization of $P(\pi)$ we provide is naturally divided into several cases, depending on which term of Equation (\star) π belongs to. First, we study the non-recursive cases, then the recursive cases with a linear root and finally the recursive cases with a prime root. We start with a preliminary study of the ways children 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 cases.

Remark B.1. *In the study that follows, we never examine the case of decomposition trees with a linear root labeled by \ominus . Indeed, permutations with decomposition trees of this form are the reverse of permutations whose decomposition trees have a linear root labeled by \oplus , and every argument and result on the \oplus case can therefore be transposed to the \ominus case. A similar symmetry holds for decomposition trees with prime roots labeled by increasing (resp. decreasing) quasi-oscillations and two children that are not leaves.*

B.1. Reading of children of a linear node

Definition B.2. *Let π be a pin-permutation and $p = (p_1, \dots, p_n)$ be a pin representation of π . We say that p reads the points of π in the order p_1, \dots, p_n . For any set D of points of π , if k is the number of maximal factors $p_i, p_{i+1}, \dots, p_{i+j}$ of p that contain only points of D , we say that D is read in k pieces by p . If C is a set of points of π disjoint from D , we say that D is read entirely before C if every pin belonging to D appears in p before the first pin belonging to C .*

Let π be a pin-permutation whose decomposition tree T has a linear root.

W.l.o.g., assume that $T = \begin{array}{c} \oplus \\ \swarrow \quad \downarrow \quad \searrow \\ T_1 \quad T_2 \quad \dots \quad T_r \end{array}$ and let $p = (p_1, \dots, p_n)$ be one of

its pin representations. In the sequel, we denote by i_0 the index of the child which contains p_1 .

Lemma B.3. *Let $1 \leq i, j \leq r$ such that either $i < j < i_0$ or $i_0 < j < i$. Then T_j is read by p entirely before T_i .*

Proof. Let $\ell = \min\{\ell', p_{\ell'} \in T_i\}$. Let $\mathcal{B}_{p_1, \dots, p_\ell}$ be the bounding box of $\{p_1, \dots, p_\ell\}$. As $p_1 \in T_{i_0}$ and $p_\ell \in T_i$, $T_j \subseteq \mathcal{B}_{p_1, \dots, p_\ell}$ (see Figure 12), hence it is entirely read before p_ℓ in p . Indeed, for all $k \geq 2$, p_k lies outside the bounding box of $\{p_1, \dots, p_{k-1}\}$. \square

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

Lemma B.4. *For every $k \in \{1, \dots, n\}$ there is at most one child whose reading has started and is not finished after (p_1, p_2, \dots, p_k) .*

Proof. Suppose that pins p_1, \dots, p_k have already been read and that there are two children T_i and T_m with $i < m$ whose readings have started and are not finished. By Lemma B.3 there exists at most one child T_j with $j < i_0$ and at most one child T_j with $j > i_0$ whose readings have started and are not finished. Therefore $i \leq i_0$ and $m \geq i_0$. 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 $\text{\textcolor{brown}{/}}/$ (see Figure 13). Such a pin is on the side of the bounding box $\mathcal{B}_{p_1, \dots, p_k}$ of $\{p_1, \dots, p_k\}$, and the same remark goes for T_m . But from Lemma 2.6 (p.7) there is at most one pin lying on the sides of a bounding box, and this contradiction concludes the proof. \square

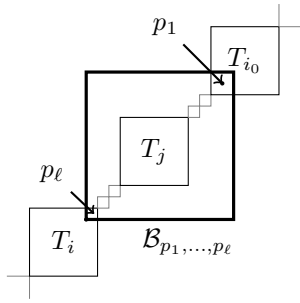


Figure 12: Proof of Lemma B.3.

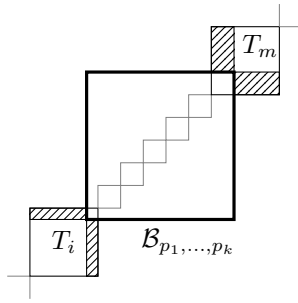


Figure 13: Proof of Lemmas B.4 and B.5.

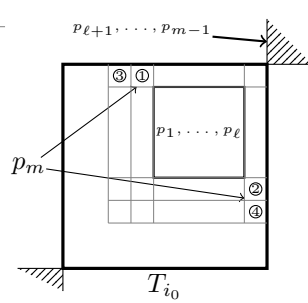


Figure 14: T_{i_0} is read in two pieces (Lemma B.6).

Lemma B.5. *Every child T_i is read in one piece by p , except perhaps T_{i_0} .*

Proof. Consider a child T_i with $i \neq i_0$ which is read in more than one piece 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 $\text{\texttt{ZZ}}$ in T_i as shown in the bottom left part of Figure 13. Since T_i is \oplus -indecomposable, there is at least one pin in this zone. This pin is on the side of the bounding box of $\{p_1, p_2, \dots, p_k\}$ so it is p_{k+1} by Lemma 2.6 (p.7). Thus $p_{k+1} \in T_i$ which provides the desired contradiction. \square

When a child may be read in several pieces, the decomposition tree of the whole permutation π has a special shape given in the following lemma.

Lemma B.6. *The only permutations π whose decomposition trees have a root \oplus in which a child may be read in several pieces are those whose decomposition trees have one of the shapes given in Figure 15 where ξ^+ is an increasing oscillation of size at least 4.*

A given permutation π may match several shapes of Figure 15. However if a child is read in more than one piece, then it is necessarily the first child to be read (denoted T_{i_0}) and it is read in two pieces; in addition, there is exactly one shape of Figure 15 such that the first part of T_{i_0} to be read is S and the second part is the remaining leaves of T_{i_0} with only the point x read in between.

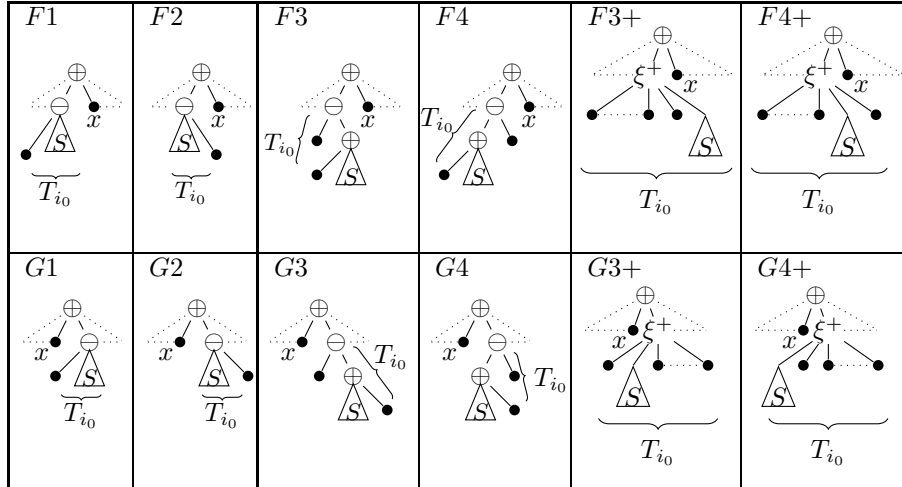


Figure 15: Decomposition tree of π when T_{i_0} may be read in several pieces.

In Figure 15 and in the sequel, we draw the attention of the reader to the difference between trees of the shape $\begin{array}{c} \text{R} \\ \swarrow \quad \searrow \\ \triangle \quad \bullet \end{array}$ and $\begin{array}{c} \text{R} \\ \swarrow \quad \searrow \\ \triangle \quad \bullet \end{array}$: in the first case the root R has exactly 2 children, in the second one it has at least two children, T being a forest.

Proof. Let π be a pin-permutation whose decomposition tree has a root \oplus . Let $p = (p_1, p_2, \dots, p_n)$ be a pin representation of π that reads one child in several

pieces. Lemma B.5 ensures that there is only one such child, which is necessarily T_{i_0} . Denote p_1, \dots, p_ℓ the first part of the reading of T_{i_0} . Then $p_{\ell+1}, \dots, p_{m-1}$ belong to other children 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 $\text{\texttt{///}}$ as shown in Figure 14. But if both zones contain at least one pin p_i with $i \in \{\ell+1, \ell+2, \dots, m-1\}$, the bounding box of $\{p_1, \dots, p_{m-1}\}$ contains T_{i_0} and thus p_m cannot be outside the bounding box of $\{p_1, \dots, p_{m-1}\}$. Hence all pins p_i with $i \in \{\ell+1, \ell+2, \dots, m-1\}$ are in the same zone.

Assume w.l.o.g. that $\{p_{\ell+1}, \dots, p_{m-1}\}$ are in the upper right zone of Figure 14 (otherwise, in the proof that follows, cases $F1, \dots, F4+$ of Figure 16 are replaced by cases $G1, \dots, G4+$). 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 child T_{i_0} which contradicts our hypothesis. Thus p_m must be a separating pin and $m = \ell + 2$.

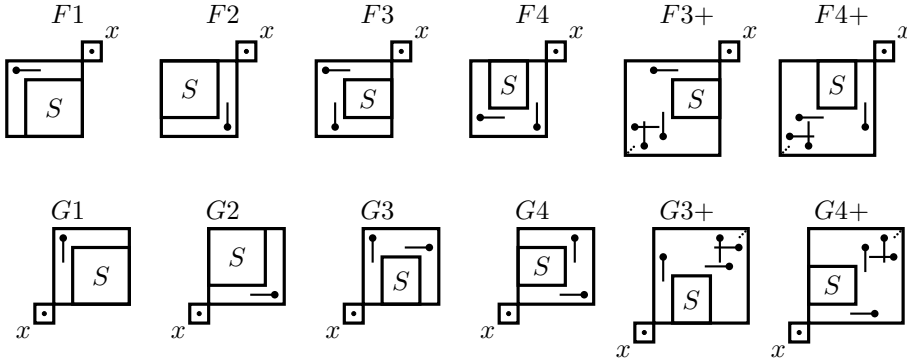


Figure 16: Diagram of T_{i_0} and x if T_{i_0} is read in two pieces, the first part being S .

As at most one point can lie on the sides of a bounding box, there are only four possible positions for p_m as depicted in Figure 14. If there is no pin separating p_m from $\{p_1, \dots, p_{m-1}\}$ then p_m is either in position ① (case $F1$ on Figure 16) or ② (case $F2$); moreover pins $\{p_1, p_2, \dots, p_\ell, p_m\}$ form a block and thus represent T_{i_0} (because T_{i_0} is \oplus -indecomposable). Otherwise there is exactly one pin p_{m+1} separating p_m from the bounding box of $\{p_1, \dots, p_\ell\}$, thus p_m is either in position ③ (cases $F3$ and $F3+$) or ④ (cases $F4$ and $F4+$). Suppose that it is in position ④ then p_{m+1} is a left pin separating p_m from the preceding ones. There are again two different cases: if p_{m+2} respects the independence condition (case $F4$), then p_{m+1} ends T_{i_0} (since T_{i_0} is \oplus -indecomposable). If p_{m+2} respects the separation condition then it can only separate p_{m+1} and $\{p_1, \dots, p_m\}$ from below (case $F4+$). This process can be repeated alternating between left and down pins until the following pin p_{m+k+1} is an independent pin, ending the child T_{i_0} .

Thus we have proved that T_{i_0} is read in exactly two pieces, p_1, \dots, p_ℓ for

the first part and p_m, \dots, p_{m+k} with $m = \ell + 2$ for the second part. And from Lemma B.5 the pin $p_{\ell+1}$ is by itself a child T_i of the root. It is then straightforward to check from Figure 16 that π has a decomposition tree of one shape given in Figure 15 with $S = \{p_1, \dots, p_\ell\}$ and $x = p_{\ell+1}$. \square

Note that in the proof of Lemma B.6, the order in which the points corresponding to the leaves of $T_{i_0} \setminus S$ are read is uniquely determined, leading to the following remark:


Remark B.7. *If a child T_{i_0} is read in two pieces with the first part fixed, then the second part consists of all remaining points of T_{i_0} and the order in which they are read is uniquely determined.*

We now start the description of the set of pin words encoding any pin-permutation, by case study on Equation (\star) (p.20).

B.2. Non-recursive cases

Permutation of size 1. The permutation $\pi = 1$ (whose decomposition tree is a leaf \bullet) has exactly four pin words – namely, $P(\pi) = \{1, 2, 3, 4\}$.

Simple permutations. The only pin-permutations whose decomposition trees have a prime root and are non-recursive are those whose decomposition trees

are of the form , i.e., the simple pin-permutations. The following theorem describes properties of their pin words.

Theorem B.8. *A simple permutation has at most 48 pin words, which are all strict or quasi-strict.*

Proof. Let π be a simple permutation. Then any pin representation p of π is proper (see Remark 2.7 p.7) and $|\pi| \geq 3$, so p_3 is a separating pin and p is associated to 6 pin words by Remark 3.2 (p.10).

Moreover by Lemmas 4.3 and 4.6 of [6] there are at most 8 possible beginnings (p_1, p_2) of a pin representation of π . Furthermore, each of these beginnings gives at most one pin representation p of π . Indeed p_{k+1} has to be the only point separating p_k from the previous points, since p_{i+1} separates p_i from previous points for all i . So π has at most 8 pin representations and at most 48 pin words. Finally the first statement of Remark 3.5 (p.11) ensures that they are all strict or quasi-strict. \square

Permutations whose decomposition trees have a linear root. W.l.o.g., since we

focus on the non-recursive case, $\pi = \xi_1 \oplus \xi_2 \dots \xi_r$ where ξ_i are increasing oscillations. Lemma B.9 is a direct consequence of Lemma B.6.

Lemma B.9. *Let $p = (p_1, p_2, \dots, p_n)$ be a pin representation of π . The only child ξ_i which may be read in several pieces is the child ξ_{i_0} to which p_1 belongs. Moreover if p reads ξ_{i_0} in several pieces, it is read in two pieces, the second child ξ_i read by p is either ξ_{i_0-1} or ξ_{i_0+1} and is a leaf, denoted x . Finally, the set $E = \xi_{i_0} \cup \{x\}$ is read in one piece by p .*

Lemma B.9 together with Lemma B.3 leads to the following.

Consequence B.10. *Every pin representation p of π begins by entirely reading two consecutive children of the root, say ξ_i and ξ_{i+1} , then p reads in one piece each of the others ξ_j . Moreover the children ξ_j for $j < i$ are read in decreasing order $(\xi_{i-1}, \xi_{i-2}, \dots, \xi_1)$ and the children ξ_j for $j > i+1$ are read in increasing order $(\xi_{i+2}, \xi_{i+3}, \dots, \xi_r)$.*

Consequence B.10 implies that the restriction of p to each child ξ_j where $j < i$ (resp. $j > i+1$) is a pin representation of ξ_j whose origin lies in quadrant 1 (resp. 3) with respect to the bounding box of the set of points of ξ_j . Indeed p_1 and p_2 are in ξ_i or ξ_{i+1} thus they lie in quadrant 1 (resp. 3) with respect to ξ_j . Since only p_2 may separate p_0 from p_1 , p_0 is also in quadrant 1 (resp. 3). This is the reason for introducing the functions $P^{(h)}$ in Subsection 4.3. Recall that for any increasing oscillation ξ , we denote by $P^{(h)}(\xi)$ the set of pin words that encode ξ and whose origin lies in quadrant $h = 1$ or 3 with respect to the points of ξ .

To characterize the pin words that encode a permutation $\pi = \oplus[\xi_1, \xi_2, \dots, \xi_r]$ where every ξ_i is an increasing oscillation, Consequence B.10 leads us naturally to introduce the shuffle product⁷ of sequences. From the above discussion, Theorem B.12 then follows, providing the desired characterization.

Definition B.11. *Let $A = (A_1, A_2, \dots, A_q)$ and $B = (B_1, B_2, \dots, B_s)$ be two sequences of sets of words. The shuffle product $A \sqcup B$ of A and B is defined as*

$$A \sqcup B = \{c = c_1 \dots c_{q+s} \mid \exists I = \{i_1, \dots, i_q\}, J = \{j_1, \dots, j_s\} \text{ with } I \cap J = \emptyset, \\ i_1 < \dots < i_q, j_1 < \dots < j_s, \text{ and } c_{i_k} \in A_k \forall 1 \leq k \leq q, c_{j_k} \in B_k \forall 1 \leq k \leq s\}.$$

For example, letting $A = (\{x\}, \{aay\}, \{aa\})$ and $B = (\{b\}, \{b, xy\})$, the shuffle of A and B is

$$\begin{array}{lll} A \sqcup B = \{x \cdot aay \cdot aa \cdot b \cdot b, & x \cdot aay \cdot aa \cdot b \cdot xy, & \text{for } \{1, 2, 3\} \uplus \{4, 5\} \\ & x \cdot aay \cdot b \cdot aa \cdot b, & x \cdot aay \cdot b \cdot aa \cdot xy, & \text{for } \{1, 2, 4\} \uplus \{3, 5\} \\ & x \cdot aay \cdot b \cdot b \cdot aa, & x \cdot aay \cdot b \cdot xy \cdot aa, & \text{for } \{1, 2, 5\} \uplus \{3, 4\} \\ & x \cdot b \cdot aay \cdot aa \cdot b, & x \cdot b \cdot aay \cdot aa \cdot xy, & \text{for } \{1, 3, 4\} \uplus \{2, 5\} \\ & x \cdot b \cdot aay \cdot b \cdot aa, & x \cdot b \cdot aay \cdot xy \cdot aa, & \text{for } \{1, 3, 5\} \uplus \{2, 4\} \\ & x \cdot b \cdot b \cdot aay \cdot aa, & x \cdot b \cdot xy \cdot aay \cdot aa, & \text{for } \{1, 4, 5\} \uplus \{2, 3\} \\ & b \cdot x \cdot aay \cdot aa \cdot b, & b \cdot x \cdot aay \cdot aa \cdot xy, & \text{for } \{2, 3, 4\} \uplus \{1, 5\} \\ & b \cdot x \cdot aay \cdot b \cdot aa, & b \cdot x \cdot aay \cdot xy \cdot aa, & \text{for } \{2, 3, 5\} \uplus \{1, 4\} \\ & b \cdot x \cdot b \cdot aay \cdot aa, & b \cdot x \cdot xy \cdot aay \cdot aa, & \text{for } \{2, 4, 5\} \uplus \{1, 3\} \\ & b \cdot b \cdot x \cdot aay \cdot aa, & b \cdot xy \cdot x \cdot aay \cdot aa\} & \text{for } \{3, 4, 5\} \uplus \{1, 2\} \end{array}$$

where every line in the above corresponds to the words of $A \sqcup B$ associated with the bipartition of $\{1, 2, 3, 4, 5\}$ into $I \uplus J$ which is indicated on the right.

⁷The shuffle product is sometimes called *merge* in the permutation patterns literature.

Theorem B.12. *The set $P(\pi)$ of pin words of a permutation $\pi = \oplus[\xi_1, \dots, \xi_r]$ where every ξ_i is an increasing oscillation is:*

$$P(\pi) = \bigcup_{1 \leq i \leq r-1} P(\oplus[\xi_i, \xi_{i+1}]) \cdot \left((P^{(1)}(\xi_{i-1}), \dots, P^{(1)}(\xi_1)) \sqcup (P^{(3)}(\xi_{i+2}), \dots, P^{(3)}(\xi_r)) \right).$$

Lemmas B.13 and B.18 below give explicit expressions for $P^{(1)}(\xi)$, $P^{(3)}(\xi)$ and $P(\oplus[\xi_i, \xi_j])$ for every increasing oscillations ξ , ξ_i and ξ_j , hence with Theorem B.12 an explicit expression for $P(\pi)$. Note that similar results can be obtained for permutations with root \ominus and decreasing oscillations using Remark B.1 (p.34).

In the following lemmas, we distinguish several cases according to the type (x, y) , with $x, y \in \{H, V\}$, of increasing oscillations – see Subsection A.2 for the definition of these types. It starts with Lemma B.13, whose proof immediately follows from a comprehensive study of the different cases illustrated in Figure 10 (p.31).

Lemma B.13. *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:

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

Remark B.14. *If the increasing oscillation ξ is of size 2 then $P^{(h)}(\xi)$ contains two words, otherwise it is a singleton. Moreover, for the map ϕ studied in Section 3 (see Definition 3.9 p.12), and for any increasing oscillation ξ , we have $\phi(P^{(3)}(\xi)) \subseteq \{U, R\}^*$ and $\phi(P^{(1)}(\xi)) \subseteq \{L, D\}^*$.*

We are further interested in describing the set of pin words of $\oplus[\xi_i, \xi_j]$ for any increasing oscillations ξ_i and ξ_j . This is achieved in Lemma B.18. For this purpose, we first describe the set $P(\xi)$ of pin words of any increasing oscillation ξ , and the set $P^{mix}(\xi_i, \xi_j)$ of pin words of $\oplus[\xi_i, \xi_j]$ such that one of the two oscillations is read in two pieces.

Lemma B.15. *Let Q^- (resp. S_H^- , resp. S_V^-) be 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): $Q^- = \{12, 14, 22, 24, 32, 34, 42, 44\}$, $S_H^- = \{1R, 2R, 3L, 4L\}$*

and $S_V^- = \{1U, 2D, 3D, 4U\}$. Define similarly Q^+ (resp. S_H^+ , resp. S_V^+) for the permutation 12.

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

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

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

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:

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

In particular, $|P(\xi)| \leq 48$ for any increasing oscillation ξ and if $|\xi| \neq 3$, $P(\xi)$ contains only strict and quasi-strict pin words.

Proof. By comprehensive examination of the cases illustrated in Figure 10. \square

Definition B.16. For any pair of increasing oscillations (ξ_i, ξ_j) , we denote by $P^{mix}(\xi_i, \xi_j)$ the set of pin words encoding a pin representation of $\oplus[\xi_i, \xi_j]$ that reads one of the two oscillations in two pieces.

Lemma B.17. Let ξ_i and ξ_j be two increasing oscillations.

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

Otherwise, assume w.l.o.g. that $\xi_i = 1$, and set $|\xi_j| = 2p + q + 1$ with $q \in \{0, 1\}$. If $|\xi_j| \geq 4$, then

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

If $|\xi_j| = 3$, i.e., $\xi_j = 231$ or 312 , we have

$$\begin{aligned} P^{mix}(1, 231) &= P(12) \cdot 3R + (13 + 23 + 33 + 43 + 1D + 4D) \cdot RU, \\ P^{mix}(1, 312) &= P(12) \cdot 3U + (13 + 23 + 33 + 43 + 1L + 2L) \cdot UR. \end{aligned}$$

If $|\xi_j| = 2$, i.e., $\xi_j = 21$, we have

$$P^{mix}(1, 21) = (13 + 23 + 33 + 43 + 1D + 4D) \cdot R + (13 + 23 + 33 + 43 + 1L + 2L) \cdot U.$$

In particular, $|P^{mix}(\xi_i, \xi_j)| \leq 22$ for any increasing oscillations ξ_i and ξ_j .

Proof. From Lemma B.9 (p.38) when one of the oscillations ξ_i or ξ_j is read in two pieces, then the other one has size 1. W.l.o.g. assume that $|\xi_i| = 1$. Then from Lemma B.6 (p.36) the decomposition tree of $\oplus[\xi_i, \xi_j]$ has one of the shapes of Figure 15 (p.36), with T_{i_0} corresponding to ξ_j and x corresponding to ξ_i .

If ξ_j has size 2, then $\xi_j = 21$ and $\oplus[\xi_i, \xi_j]$ maps only to configurations $G1$ and $G2$ (see Figures 15 and 16). Therefore there are two pin representations of $\oplus[\xi_i, \xi_j]$ where ξ_j is read in two pieces. The twelve corresponding pin words (see Remark 3.2 p.10) are those given in Lemma B.17.

If ξ_j has size 3, then $\xi_j = 231$ (resp. 312) and $\oplus[\xi_i, \xi_j]$ maps only to configurations $G2$ and $G4$ (resp. $G1$ and $G3$), and we conclude similarly.

Otherwise, $|\xi_j| \geq 4$. Since ξ_j is an increasing oscillation, $\oplus[\xi_i, \xi_j]$ maps only to configuration $G3+$ or to configuration $G4+$, with $|S| = 1$. These two cases are exclusive, and the configuration ($G3+$ or $G4+$) to which $\oplus[\xi_i, \xi_j]$ maps is determined by the type (V or H) of the lower left active knight of ξ_j . Lemma B.6 and Remark B.7 (p.38) ensure that there is exactly one pin representation for $\oplus[\xi_i, \xi_j]$ that reads ξ_j in two pieces. The six corresponding pin words are those given in Lemma B.17. \square

From the expressions of $P, P^{mix}, P^{(1)}$ and $P^{(3)}$ we can deduce the explicit expression of $P(\oplus[\xi_i, \xi_j])$ making use of the following result:

Lemma B.18. *For any pair of increasing oscillations (ξ_i, ξ_j) :*

- If $|\xi_i| > 1$ and $|\xi_j| > 1$, $P(\oplus[\xi_i, \xi_j]) = (P(\xi_j) \cdot P^{(1)}(\xi_i)) \cup (P(\xi_i) \cdot P^{(3)}(\xi_j))$
- If $|\xi_i| = 1$ and $|\xi_j| = 1$, $P(\oplus[\xi_i, \xi_j]) = P(12)$
- Otherwise assume w.l.o.g. that $|\xi_i| = 1$ and $|\xi_j| = 2p + q$ with $q \in \{0, 1\}$:

$$P(\oplus[\xi_i, \xi_j]) = P^{mix}(\xi_i, \xi_j) \bigcup (P(\xi_j) \cdot 3) \bigcup (\{1, 2, 3, 4\} \cdot P^{(3)}(\xi_j)) \bigcup P^{sep}(\xi_j)$$

$$\text{with } P^{sep}(\xi_j) = \begin{cases} (2+3) \cdot (UR)^p U^q & \text{if } \xi_j \text{ has type } (H, H) \text{ or } (H, V) \\ (3+4) \cdot (RU)^p R^q & \text{if } \xi_j \text{ has type } (V, V) \text{ or } (V, H). \end{cases}$$

In particular, $|P(\oplus[\xi_i, \xi_j])| \leq 192$ for any oscillations ξ_i and ξ_j .

Proof. For the first item, Lemma B.17 ensures that the pin words of $\oplus[\xi_i, \xi_j]$ encode pin representations reading both ξ_i and ξ_j in one piece. The two terms of the union are obtained according to which oscillation (among ξ_i and ξ_j) is read first. The second statement follows directly from the fact that $\oplus[\xi_i, \xi_j] = 12$ in this case. From Lemma B.17, the situation of the third statement is the one where there are pin words encoding pin representations reading ξ_j in two pieces. The four terms of the union account for four different kinds of pin words. Namely, $P^{mix}(\xi_i, \xi_j)$ is the set of pin words reading ξ_j in two pieces, $P(\xi_j) \cdot 3$ is the set of pin words reading first ξ_j and then ξ_i , $\{1, 2, 3, 4\} \cdot P^{(3)}(\xi_j)$ is the set of pin words reading first ξ_i and then ξ_j starting with an independent pin, and $P^{sep}(\xi_j)$ is the set of pin words reading first ξ_i and then ξ_j starting with

a separating pin. Notice that in this last situation, the first pin of ξ_j may be separating only because $|\xi_i| = 1$, so that this case does not need to appear in the first item. \square

This completes the explicit description of all the sets of pin words appearing in Theorem B.12.

B.3. Recursive case: decomposition trees with a linear root

We now focus on pin-permutations whose decomposition trees have a linear root \oplus and a child T_{i_0} which is not an increasing oscillation. From [6, Lemma 3.7] T_{i_0} is then the first child read by any pin representation. Lemma B.6 (p.36) gives a characterization of permutations in which T_{i_0} may be read in several pieces. Moreover from Remark B.7 if T_{i_0} is read in two pieces the first part S being fixed, then the order of the points of the remaining part is uniquely determined. Nevertheless, since some permutations may satisfy several conditions $F1$ to $G4+$ of Lemma B.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 $G2$ (see Figure 16 p.37). In Figure 17 we classify the permutations according to the conditions they satisfy.

Let \mathcal{H} be the set of permutations in which T_{i_0} may be read in several pieces. Then any permutation of \mathcal{H} satisfies exactly one of the conditions (iHj) of Figure 17. We say that a permutation satisfies condition (iHj) when its diagram has the corresponding shape in Figure 17 (up to symmetry) and does not satisfy any condition that appears above (iHj) in Figure 17. For example a permutation in $(1H2)$ cannot be in $(2H2)$. One can check by a comprehensive verification that there is no other combination (up to symmetry) of the conditions $F1$ to $G4+$ of Lemma B.6. Moreover as T_{i_0} is not an increasing oscillation, the sets S and T that appear on Figure 17 are such that $|S| \geq 2$ and $|T| \geq 1$.

Similarly to $P(\pi)$ denoting the set of pin words encoding π , we denote by $P(T)$ the set of pin words that encode the permutation whose decomposition tree is T .

Theorem B.19. *Let $\pi = \xi_1 \dots \xi_\ell \dots \xi_{\ell+2} \dots \xi_r$ be a \oplus -decomposable permutation where T_{i_0} is the only child that is not an increasing oscillation.*

For every i such that $1 \leq i \leq \ell$ and j such that $\ell + 2 \leq j \leq r$, set

$$\mathfrak{P}_{(i)}^{(1)} = (P^{(1)}(\xi_i), \dots, P^{(1)}(\xi_1)) \text{ and } \mathfrak{P}_{(j)}^{(3)} = (P^{(3)}(\xi_j), \dots, P^{(3)}(\xi_r)).$$

We describe below the set $P(\pi)$ of pin words encoding π . When $\pi \in \mathcal{H}$, these sets are given only when the diagram of π is one of those shown in Figure 17. When the diagram of π is one of their symmetries, $P(\pi)$ is modified accordingly.

• *If $\pi \notin \mathcal{H}$ (i.e., if π does not satisfy any condition shown up to symmetry on Figure 17) then $P(\pi) = P_0 = P(T_{i_0}) \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$.*

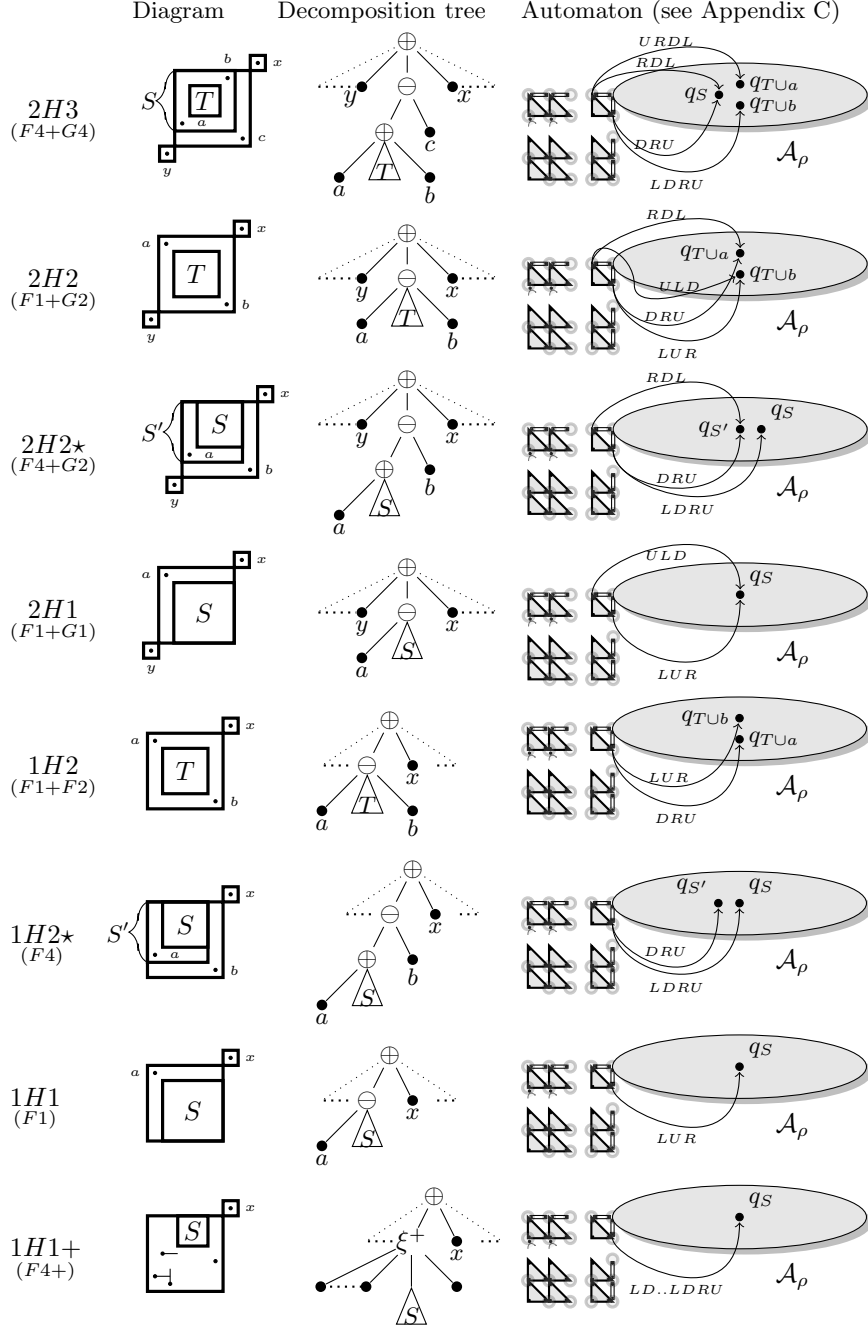


Figure 17: The set \mathcal{H} and conditions (iHj): $\pi \in \mathcal{H}$ if and only if π satisfies one of the conditions (iHj) shown above up to symmetry, that form a partition of \mathcal{H} .

- If π satisfies condition (1H1) then $P(\pi) = P_0 \cup P_1$, 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 π satisfies condition (1H1+) then $P(\pi) = P_0 \cup P_1$, 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} . Notice that w is obtained from the unique word of $P^{(1)}(\xi)$ (see Remark B.14 p.40) by deleting its first letter.

- If π satisfies condition (1H2 \star) then $P(\pi) = P_0 \cup P_1 \cup P_2$, with

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

- If π satisfies condition (1H2) then $P(\pi) = P_0 \cup P_1 \cup P_2$, with

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

- If π satisfies condition (2H1) then $P(\pi) = P_0 \cup P_1 \cup P_2$, with

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

- If π satisfies condition (2H2 \star) then $P(\pi) = P_0 \cup P_1 \cup P_2 \cup P_3$, with

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

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

- If π satisfies condition (2H2) then $P(\pi) = P_0 \cup P_1 \cup P_2 \cup P_3 \cup P_4$, with

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

$$\begin{aligned}
P_3 &= P(T \cup a) \cdot \underbrace{3 \cdot R}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}, \quad P_4 = P(T \cup b) \cdot \underbrace{3 \cdot U}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}. \\
&\bullet \text{ If } \pi \text{ satisfies condition (2H3) then } P(\pi) = P_0 \cup P_1 \cup P_2 \cup P_3 \cup P_4, \text{ with} \\
P_1 &= P(S) \cdot \underbrace{1 \cdot D}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}, \quad P_2 = P(T \cup b) \cdot \underbrace{1 \cdot D \cdot L}_{x \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+3)}^{(3)}, \\
P_3 &= P(T \cup a) \cdot \underbrace{3 \cdot R \cdot U}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}, \quad P_4 = P(S) \cdot \underbrace{3 \cdot R}_{y \cup T_{i_0}} \cdot \mathfrak{P}_{(\ell-1)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}.
\end{aligned}$$

Proof. For each item, it is easy to check that the given pin words are pin words encoding π . Conversely, we prove that a pin word encoding π is necessarily in the set claimed to be $P(\pi)$. First of all, by [6, Lemma 3.7], every pin representation of π starts in the only child that is not an increasing oscillation, *i.e.*, with T_{i_0} .

Let us start with the first point of Theorem B.19. In this case, by definition of \mathcal{H} , we know that T_{i_0} is read in one piece. By Lemma B.5 (p.35), the other children are also read in one piece, and Lemma B.3 ensures that the children closest to T_{i_0} are read first. As there is no relative order between children $\xi_{\ell+2}$ to ξ_r and children ξ_ℓ to ξ_1 , this leads to the shuffling operation between pin words corresponding to these children, with an external origin placed in quadrant 3 (resp. 1) with respect to their bounding box.

In the other cases of Theorem B.19, by Lemma B.6, every pin representation of π either reads T_{i_0} in one piece or in two pieces. In case T_{i_0} is read in one piece, the pin representation is as before encoded by pin words of P_0 . If it is read in two pieces, Lemma B.6 and its proof and Remark B.7 ensure that the corresponding pin words are those described.

Consider for example π satisfying condition (1H1). Then π satisfies condition F1 of Lemma B.6, and only this one by definition of (1H1). If T_{i_0} is read in two pieces, then Lemma B.6 ensures that S is the first part of $T_{i_0} \cup \{x\}$ to be read, followed by x and finally a . The corresponding pin words are indeed those described in P_1 .

Taking the other example of 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 G4 with $S = T \cup a$ and P_4 corresponds to condition G2 with $S = T \cup a \cup b$. \square

Remark B.20. If π is a \ominus -decomposable permutation, a similar description of $P(\pi)$ can be obtained from Remark B.1 (p.34).

B.4. Recursive case: decomposition trees with a prime root

We now turn to the study of the recursive case where the decomposition tree has a root which is a simple permutation α . We start with the case where

$\pi = \bullet \text{---} \alpha \text{---} \bullet$ for a tree T that is not a leaf.

We begin with the characterization of the possible ways a pin representation of π may read T , introducing first a condition that will be useful in the sequel.

Definition B.21. For a permutation $\pi = \alpha[1, \dots, 1, T, 1, \dots, 1]$ with $\alpha = \alpha_1 \dots \alpha_k$, we define condition (C) as follows:

$$(C) \begin{cases} \bullet \alpha \text{ is an increasing - resp. decreasing - quasi-oscillation (see p.32);} \\ \bullet T \text{ expands an auxiliary point of } \alpha; \\ \bullet \text{ the shape of } T \text{ is } \begin{array}{c} \oplus \\ \triangle \\ T' \end{array} \text{ - resp. } \begin{array}{c} \ominus \\ \triangle \\ T' \end{array} \text{ - if the auxiliary point is } \alpha_k \\ \text{or } \alpha_{k-1} \text{ and } \begin{array}{c} \oplus \\ \triangle \\ T' \end{array} \text{ - resp. } \begin{array}{c} \ominus \\ \triangle \\ T' \end{array} \text{ - if the auxiliary point is } \alpha_1 \text{ or } \alpha_2. \end{cases}$$

Lemma B.22. Let $p = (p_1, \dots, p_n)$ be a pin representation associated to $\pi = \alpha[1, \dots, 1, T, 1, \dots, 1]$. Then one of the following statements holds:

- (1) $p_1 \in T$ and T is read in one piece by p ;
- (2) $T = \{p_1, \dots, p_i\} \cup \{p_n\}$ with $i \neq n-1$, and π satisfies condition (C);
- (3) $p_1 \notin T$, $T = \{p_2, p_n\}$, and π satisfies condition (C).

Moreover if (3) is satisfied then p is a proper pin representation uniquely determined by α and its auxiliary point; it is up to symmetry the one depicted in the first diagram of Figure 18. If (2) is satisfied, defining T' as in condition (C), then $T' = \{p_1, \dots, p_i\}$ and (p_{i+1}, \dots, p_n) is uniquely determined by α and its auxiliary point, as shown in the second diagram of Figure 18 up to symmetry.

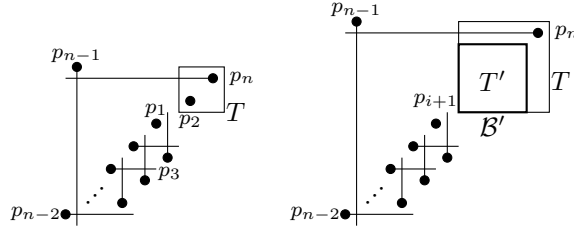


Figure 18: Diagram of π when one child T is not a leaf, is read in two pieces and $p_1 \notin T$ or $p_1 \in T$.

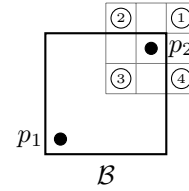


Figure 19: Possible positions for p_n .

Proof. If $p_1 \notin T$, then by Lemma 3.12(ii) of [6], $T = \{p_2, p_n\}$. Up to symmetry assume that $\{p_1, p_2\}$ is an increasing subsequence of π . As $\{p_2, p_n\}$ forms a block, p_n is in one of the 4 positions shown in Figure 19. 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, if p_n lies in one of these positions, it must be read immediately after p_1 and p_2 and thus must be p_3 from Lemma 2.6 (p.7). But $n > 3$ (α is simple so $|\alpha| \geq 4$) so that these positions are also forbidden. Hence p_n lies in position ① and $T = 12$.

As α is a simple pin-permutation, p_3 respects the separation condition. But if p_3 lies above or on the right of the bounding box \mathcal{B} 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 $|\alpha| = 3$, contradicting the fact that α is simple.

By symmetry we can assume that p_3 lies below \mathcal{B} (see the first diagram of Figure 18). 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 pin (depending on the parity of n). Then α is a quasi-oscillation in which the point expanded by T is an auxiliary point and $T = 12$ or $T = 21$ depending on the nature of α – increasing or decreasing. Consequently, π satisfies condition (C). Notice that given α and its auxiliary point, once we know that $p_1 \notin T$ then p is uniquely determined.

Suppose now that $p_1 \in T$ but T is not read in one piece. By Lemma 3.11(i) of [6], it is read in two pieces, 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 sides of the bounding box \mathcal{B}' of $\{p_1, \dots, p_i\}$. Up to symmetry, we can assume that p_n lies in quadrant 1 with respect to \mathcal{B}' (see Figure 18). Therefore, $T = \begin{array}{c} \oplus \\ \triangle \\ T' \end{array}$, T' being the sub-forest of T whose leaves are the points in \mathcal{B}' , i.e., are p_1, \dots, p_i . As T is a block, no pin must lie on the sides of the bounding box of $\{p_1, \dots, p_i, p_n\}$. Moreover p_{i+1} does not lie in quadrant 1 with respect to T , otherwise p_n would lie inside the bounding box of $\{p_1, \dots, p_{i+1}\}$. 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 α being simple. Thus p_{i+1} lies in quadrant 3 with respect to T . The same goes for p_j with $j \in \{i+1, \dots, n-2\}$. Because α is simple, we therefore deduce that all 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 an auxiliary point. Moreover, π satisfies condition (C) with $T' = \{p_1, \dots, p_i\}$, as can be seen (up to symmetry) on the right part of Figure 18. Notice that given α and its auxiliary point, once we know that $T = \{p_1, \dots, p_i\} \cup \{p_n\}$ with $i \neq n-1$ then (p_{i+1}, \dots, p_n) is uniquely determined.

Finally if we are not in one of the two cases discussed above, then $p_1 \in T$ and T is read in one piece by p , concluding the proof. \square

With the description of the pin representations of π in Lemma B.22, we are able to give in Theorem B.25 below an explicit description of the set $P(\pi)$ of pin words that encode π . The statement of Theorem B.25 makes use of the notation $Q_x(\alpha)$, which has appeared in Subsection 4.3, and that we define below.

Definition B.23. *For every simple pin-permutation α , with an active point x (see p.30) marked, we define $Q_x(\alpha)$ as the set of strict pin words obtained by deleting the first letter of a quasi-strict pin word of α whose first point read in α is x .*

Notice that $|u| = |\alpha| - 1$ for all $u \in Q_x(\alpha)$.

Remark B.24. To each pin representation of α whose first point read is x corresponds exactly one word of $Q_x(\alpha)$. Indeed the quasi-strict pin words associated to a pin representation differ only in their first letter (see Figure 6 and Remark 3.2 p.10).

Theorem B.25. Let π be a pin-permutation, whose decomposition tree has a prime root α , with exactly one child T that is not a leaf. Then, denoting by x the point of α expanded by T , the following holds:

- If π does not satisfy condition (C), then $P(\pi) = P(T) \cdot Q_x(\alpha)$.
- If π satisfies condition (C), we define T' as in (C), and we distinguish two sub-cases according to the number of leaves $|T|$ of T :
 - (a) if $|T| \geq 3$, let w be the unique word encoding the unique reading of the remaining leaves in π after T' is read when T is read in two pieces. Then $P(\pi) = P(T) \cdot Q_x(\alpha) \cup P(T') \cdot w$.
 - (b) if $|T| = 2$, let $P_{\{1,n\}}(\pi)$ be the set of pin words encoding the unique pin representation p of π such that $T = \{p_1, p_n\}$. Define similarly $P_{\{2,n\}}(\pi)$ for the case $T = \{p_2, p_n\}$. Then $P(\pi) = P(T) \cdot Q_x(\alpha) \cup P_{\{1,n\}}(\pi) \cup P_{\{2,n\}}(\pi)$.

Proof. In each case, it is easy to check that the given pin words are pin words encoding π . Conversely, we prove that a pin word encoding π is necessarily in the set claimed to be $P(\pi)$. Let $u = u_1 \dots u_n \in P(\pi)$ and $p = (p_1, \dots, p_n)$ be the associated pin representation. Then p satisfies one statement of Lemma B.22.

If p satisfies statement (1) of Lemma B.22 then, setting $k = |T|$, (p_k, \dots, p_n) is a pin representation of α beginning with x . Moreover as T is a block of π , p_{k+1} is an independent pin, so that u_{k+1} is a numeral. Thus for all h in $\{1, 2, 3, 4\}$, $h u_{k+1} \dots u_n$ is a pin word encoding α and starting with two numerals. As α is simple, its pin words are strict or quasi-strict, hence $h u_{k+1} \dots u_n$ is quasi-strict. Therefore $u_{k+1} \dots u_n \in Q_x(\alpha)$. Moreover (p_1, \dots, p_k) is a pin representation of T . Hence $u \in P(T) \cdot Q_x(\alpha)$ which is included in the set claimed to be $P(\pi)$, regardless of whether π satisfies condition (C) or not.

If p satisfies statement (2) of Lemma B.22 then π satisfies condition (C). If $|T| = 2$ then $T = \{p_1, p_n\}$ and $u \in P_{\{1,n\}}(\pi)$. Notice that the uniqueness of the pin representation such that $T = \{p_1, p_n\}$ follows from Lemma B.22. Indeed in this case (p_{i+1}, \dots, p_n) is uniquely determined, $i = 1$ and p_1 is the only remaining point. If $|T| \geq 3$ then from Lemma B.22, $T' = \{p_1, \dots, p_{k-1}\}$ with $k = |T|$ thus the prefix of length $k - 1$ of u is in $P(T')$. From Lemma B.22, (p_k, \dots, p_n) is uniquely determined. Moreover, as $k \geq 3$, Remark 3.2 (p.10) ensures that the letters encoding these points are uniquely determined. This allows to define uniquely the word w encoding (p_k, \dots, p_n) , yielding $u \in P(T') \cdot w$.

If p satisfies statement (3) of Lemma B.22 then π satisfies condition (C), $|T| = 2$ and $u \in P_{\{2,n\}}(\pi)$. Notice that the uniqueness of the pin representation such that $T = \{p_2, p_n\}$ follows from Lemma B.22. \square

To make the set $P(\pi)$ of pin words in the statement of Theorem B.25 explicit (up to the recursive parts $P(T)$ and $P(T')$), we conclude the study of the case $\pi = \alpha[1, \dots, 1, T, 1, \dots, 1]$ by stating some properties of the (sets of) words $Q_x(\alpha)$, w , $P_{\{1,n\}}(\pi)$ and $P_{\{2,n\}}(\pi)$ that appear in Theorem B.25.

Remark B.26. *The set $Q_x(\alpha) \subseteq P(\alpha)$ can be determined in linear time w.r.t. $|\alpha|$. Indeed as α is simple it is sufficient to examine the proper pin representations of α which start with an active knight containing x . By Lemma 2.6 (p.7), these are entirely determined by their first two points. Since α is simple these two points are in knight position. Consequently, there are at most 8 proper pin representations of α starting with x , and associated pin words are obtained in linear time using Remark 3.2 (p.10).*

Lemma B.27. *In Theorem B.25, when π satisfies condition (C) and $|T| \geq 3$, the word w is a strict pin word of length at least 4 encoding α . Denoting by w' the suffix of length 2 of w , then $P(T') \cdot \phi^{-1}(w') \subseteq P(T)$. Moreover there exist a word \bar{w} of $Q_x(\alpha)$ and a letter Z such that $w = \bar{w} \cdot Z$ and no word of $\phi(Q_x(\alpha))$ contains Z . Finally when $|\alpha| \geq 5$ then $Q_x(\alpha)$ contains only \bar{w} . Otherwise $|\alpha| = 4$ and $Q_x(\alpha)$ contains two words.*

Proof. Assume that π satisfies condition (C) and $|T| \geq 3$. Define T' as in condition (C) and let $i = |T'|$. Notice that $i \geq 2$. By definition of w there exists a pin representation $p = (p_1, \dots, p_n)$ of π such that $T' = \{p_1, \dots, p_i\}$, T is read in two pieces and any corresponding pin word $u = u_1 \dots u_n$ satisfies $u_{i+1} \dots u_n = w$. Then p satisfies statement (2) of Lemma B.22, thus $T = \{p_1, \dots, p_i\} \cup \{p_n\}$ as in the second diagram of Figure 18 and $\{p_{i+2}, \dots, p_n\}$ are separating pin. As $i \geq 2$ and T' is a block of π , p_{i+1} is an independent pin encoded with a numeral. So $w = u_{i+1} \dots u_n$ is a strict pin word.

Moreover as $T = \{p_1, \dots, p_i\} \cup \{p_n\}$, (p_{i+1}, \dots, p_n) is a pin representation of α ending with x thus w is a pin word encoding α and $|w| = |\alpha| \geq 4$. Likewise (p_i, \dots, p_{n-1}) is a pin representation of α beginning with x .

Denoting by w' the suffix of length 2 of w , then from Lemma 3.10 (p.12) $\phi^{-1}(w')$ is a numeral indicating the quadrant in which p_n lies with respect to T' . And as $T = T' \cup \{p_n\}$, for all u' in $P(T')$, $u' \cdot \phi^{-1}(w')$ belongs to $P(T)$.

Moreover letting \bar{w} be the prefix of w of length $|w| - 1$, for all h in $\{1, 2, 3, 4\}$, $h\bar{w}$ is a quasi-strict pin word of α . Therefore $\bar{w} \in Q_x(\alpha)$. Denoting Z the last letter of w , Z encodes p_n . Moreover \bar{w} encodes p_{i+1}, \dots, p_{n-1} and the position of p_{i+1}, \dots, p_n is the same (up to symmetry) as the one shown on Figure 18. On this figure, it is immediate to check that $\phi(\bar{w})$ does not contain Z . To prove that this holds not only for \bar{w} but also for all words of $Q_x(\alpha)$, we first study the cardinality of $Q_x(\alpha)$.

From Remark B.24, to each pin representation of α whose first point read is x corresponds exactly one word of $Q_x(\alpha)$. Recall from Remark B.26 that a pin representation of α is determined by its first two points, which form an active knight. So we just have to compute the number of active knights of α to which x belongs, remembering that α is an increasing quasi-oscillation and x is an auxiliary point of α . This question has been addressed in Remark A.3 (p.33).

It follows that if α is an increasing quasi-oscillation of size greater than 4 and x is an auxiliary point of α , then $|Q_x(\alpha)| = 1$ as x belongs to only one active knight; and when $|\alpha| = 4$, $|Q_x(\alpha)| = 2$ as x belongs to two active knights.

To conclude the proof, recall that the word \bar{w} defined earlier belongs to $Q_x(\alpha)$ and is such that $\phi(\bar{w})$ does not contain Z . When $|\alpha| \neq 4$, we have $|Q_x(\alpha)| = 1$ so that $Q_x(\alpha) = \{\bar{w}\}$ and we conclude that no word of $\phi(Q_x(\alpha))$ contains Z . When $|\alpha| = 4$, $|Q_x(\alpha)| = 2$ and there is only one word \bar{w}' different from \bar{w} in $Q_x(\alpha)$, which may be computed from Figure 11 (p.33). We then check by comprehensive verification (of the four cases of size 4 on Figure 11) that $\phi(\bar{w}')$ does not contain Z . Details are left to the reader. \square

We are furthermore able to describe w explicitly in Remark B.29 below, and we record its expression here for future use in our work.

Definition B.28. *To each quasi-oscillation α of which an auxiliary point A is marked, we associate a word w_α^A defined below. Denoting by M the main substitution point of α corresponding to A and by $K_{A,M}$ the active knight formed by A and M then:*

When α is increasing and $K_{A,M}$ is of type H (resp. V),

if A is in the top right corner of α , we set

$$\begin{aligned} w_\alpha^A &= (DL)^{p-2}DRU \text{ (resp. } w_\alpha^A = (LD)^{p-2}LUR) \text{ if } |\alpha| = 2p \\ w_\alpha^A &= (DL)^{p-2}UR \text{ (resp. } w_\alpha^A = (LD)^{p-2}RU) \text{ if } |\alpha| = 2p-1; \end{aligned}$$

if A is in the bottom left corner of α , we set

$$\begin{aligned} w_\alpha^A &= (UR)^{p-2}ULD \text{ (resp. } w_\alpha^A = (RU)^{p-2}RDL) \text{ if } |\alpha| = 2p \\ w_\alpha^A &= (UR)^{p-2}DL \text{ (resp. } w_\alpha^A = (RU)^{p-2}LD) \text{ if } |\alpha| = 2p-1; \end{aligned}$$

When α is decreasing, w_α^A is obtained by symmetry exchanging left and right.

Notice that for quasi-oscillations that are both increasing and decreasing the choice of A determines their nature, so that w_α^A is properly defined.

Remark B.29. *If A is in the top right corner of α (see Figure 18 p.47 or Figure 9 p.31), then $w = 3 \cdot w_\alpha^A$. If A is in the bottom left (resp. top left, bottom right) corner of α then $w = 1 \cdot w_\alpha^A$ (resp. $w = 4 \cdot w_\alpha^A$, $w = 2 \cdot w_\alpha^A$).*

Remark B.30. *In Theorem B.25, if π satisfies condition (C) and $|T| = 2$, then $P_{\{1,n\}}(\pi) \cup P_{\{2,n\}}(\pi) = \{u \in P(\pi) \mid u \text{ strict or quasi-strict}\}$, denoted $P_{\text{sqS}}(\pi)$. This set corresponds to two proper pin representations, so it contains 12 pin words (see Remark 3.2 p.10). Moreover, with the notations of Lemma B.15 (p.40), and $K_{A,M}$ as in Definition B.28, we have an explicit description of $P_{\text{sqS}}(\pi)$:*

When $K_{A,M}$ is of type H (resp. V),

if α is increasing (see Figure 18 p.47), then

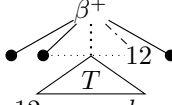
$$P_{\text{sqS}}(\pi) = (Q^+ + S_H^+) \cdot w_\alpha^A \text{ (resp. } P_{\text{sqS}}(\pi) = (Q^+ + S_V^+) \cdot w_\alpha^A)$$

and if α is decreasing, then

$$P_{\text{sqS}}(\pi) = (Q^- + S_H^-) \cdot w_\alpha^A \text{ (resp. } P_{\text{sqS}}(\pi) = (Q^- + S_V^-) \cdot w_\alpha^A).$$

This concludes the study of the case $\pi = \alpha[1, \dots, 1, T, 1, \dots, 1]$. It now remains to deal with the case where more than one child of α is not a leaf.

From Theorem 3.1 of [6] (see also Equation (\star) p.20), in this case α is an increasing (resp. decreasing) quasi-oscillation having exactly two children that are not leaves, and these are completely determined.

Theorem B.31. Let $\pi =$  where β^+ is an increasing quasi-

oscillation, the permutation 12 expands an auxiliary point of β^+ and T (of size at least 2) expands the corresponding main substitution point of β^+ . Then $P(\pi) = P(T) \cdot w$ where w is the unique word encoding the unique reading of the remaining leaves in π after T is read.

Proof. Let $p = (p_1, \dots, p_n)$ be a pin representation associated to π . According to Section 3.4 (and more precisely Figure 10) of [6], the configuration depicted on Figure 20 is the only possible configuration up to symmetry for a pin-permutation whose root is a simple permutation with two non trivial children. Thus the sequence (p_{k+1}, \dots, p_n) is uniquely determined in π . Moreover $k+1 \geq 3$, so that the suffix encoding (p_{k+1}, \dots, p_n) in a pin word of p is a word w uniquely determined from Remark 3.2 (p.10). \square

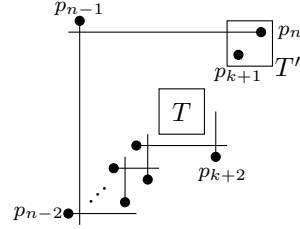


Figure 20: Diagram of π if two children are not leaves

Remark B.32. The word w in the statement of Theorem B.31 is a strict pin word uniquely determined by β^+ and the two points expanded in β^+ .

More precisely, taking the notations of Definition B.28 (with β^+ instead of α), $w = 1 \cdot w_{\beta^+}^A$ (resp. $w = 3 \cdot w_{\beta^+}^A$) when A is in the top right (resp. bottom left) corner of β^+ (see Figure 20).

For decomposition trees whose root is a decreasing quasi-oscillation, we obtain from Remark B.1 (p.34) a description of $P(\pi)$ similar to the one of Theorem B.31.

C. Building deterministic automata \mathcal{A}_π accepting the languages $\overleftarrow{\mathcal{L}}_\pi$

Appendix B gives a recursive description of the set $P(\pi)$ of pin words encoding π , for any pin-permutation π . As explained in Subsection 4.3, we next use this precise knowledge about $P(\pi)$ to build deterministic automata \mathcal{A}_π recognizing the languages $\overleftarrow{\mathcal{L}}_\pi$, for any pin-permutation π . Recall from Subsection 4.3 (p.19) that

$$\overleftarrow{\mathcal{L}}_\pi = \bigcup_{\substack{u \in P(\pi) \\ u = u^{(1)} u^{(2)} \dots u^{(j)}}} A^* \overleftarrow{\phi(u^{(j)})} A^* \dots A^* \overleftarrow{\phi(u^{(2)})} A^* \overleftarrow{\phi(u^{(1)})} A^*$$

where $A = \{U, D, L, R\}$, ϕ is the map introduced in Definition 3.9 (p.12) and for every pin word u , by $u = u^{(1)}u^{(2)} \dots u^{(j)}$ we mean (here and everywhere after) that $u^{(1)}u^{(2)} \dots u^{(j)}$ is the strong numeral-led factor decomposition of u .

To build these automata \mathcal{A}_π recognizing $\overleftarrow{\mathcal{L}}_\pi$, we proceed again recursively, distinguishing several cases following Equation (\star) p.20, *i.e.*, according to the shape of the decomposition tree of π . We also present an alternative construction of \mathcal{A}_π whose complexity is optimized; but instead of $\overleftarrow{\mathcal{L}}_\pi$, the automaton recognizes a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$. In both constructions, the automata \mathcal{A}_π are deterministic, and we have explained in Subsection 4.2 that this is the key to control the complexity of our algorithm. Moreover, in addition to being deterministic, both automata are complete and have a unique final state without outgoing transitions except for a loop labeled by all letters of A . These properties of \mathcal{A}_π are inherited from the smaller automata used in its construction.

Our construction of automata accepting words $v \in \overleftarrow{\mathcal{L}}_\pi$ relies on a greedy principle: at each step we find the first occurrence of $\phi(u^{(\ell)})$ that appears in the suffix of the word v that has not yet been read by the automaton. This is facilitated by the fact that in $\overleftarrow{\mathcal{L}}_\pi$ the factors $\phi(u^{(\ell)})$ are separated by A^* .

The reason why we consider *reversed* words is in order to preserve determinism. Indeed intuitively the possible beginnings of pin words encoding a permutation may be numerous, whereas all these words end with very similar shuffle products as it appears in Theorems B.12 and B.19 (p.40 and 43).

The description of our construction of the automata \mathcal{A}_π is organized as follows. In Subsection C.1, we present generic constructions of automata that will be used several times. In Subsections C.2 to C.5, we construct recursively the automata \mathcal{A}_π that recognize the languages $\overleftarrow{\mathcal{L}}_\pi$ for any pin-permutation π , distinguishing cases according to the decomposition tree of π – see Equation (\star) p.20. In these constructions, some states of the automata must be marked, and this is detailed in Subsection C.6. We conclude with Subsection C.7 that analyzes the complexity of building \mathcal{A}_π .

C.1. Generic constructions of deterministic automata

We present some generic constructions that are used in the next subsections. We refer the reader to [16] for more details about automata.

Aho-Corasick algorithm. Let X be a finite set of words over a finite alphabet A . The Aho-Corasick algorithm [1] builds a deterministic automaton that recognizes A^*X in linear time and space w.r.t. the sum $\|X\|$ of the lengths of the words of X . The first step of the algorithm consists in constructing a tree-automaton whose states are labeled by the prefixes of the words of X . The initial state is the empty word ε . For any word u and any letter a there is a transition labeled by a from state u to state ua if ua is a prefix of a word of X . At this step the final states are the leaves of the tree. The second step consists in adding transitions in the automaton according to a breadth-first traversal of the tree-automaton to obtain a complete automaton. For any state u and any

letter a , the transition from u labeled by a goes to the state corresponding to the longest suffix of ua that is also a prefix of a word of X . The set of final states is the set of states corresponding to words having a suffix in X . These states correspond to a leaf or an internal node – when there is a factor relation between two words of X – of the original tree-automaton. The ones corresponding to internal nodes are marked on the fly during the construction of the missing transitions.

Remark C.1. *Notice that all transitions labeled with a letter of A that does not appear in any word of X go to the initial state. Moreover the reading of any word u by the automaton leads to the state labeled with the longest suffix of u that is also a prefix of a word of X .*

A variant for first occurrences. An adaptation of the Aho-Corasick algorithm allows us to build in linear time and space w.r.t. $\|X\|$ a deterministic automaton, denoted $\mathcal{AC}(X)$, recognizing the set of words ending with a *first* occurrence of a word of X (which is strictly included in A^*X). First we perform the first step of the Aho-Corasick algorithm on X , obtaining a tree automaton. We modify the second step as follows: in the breadth-first traversal, we stop the exploration of a branch and delete its descendants as soon as a final state is reached. Moreover we do not build the outgoing transitions from the final states, nor the loops on the final states. This ensures that the language recognized is the set of words ending with a first occurrence of a word of X . Finally we merge the final states into a unique final state f to obtain $\mathcal{AC}(X)$. Moreover if we add a loop labeled by all letters of A on f we obtain an automaton $\mathcal{AC}^\circ(X)$ that recognizes the set A^*XA^* of words having a factor in X .

Remark C.2. *The main difference between our variant and the construction of Aho-Corasick is that we stop as soon as a first occurrence of a word of X is read. This ensures that $\mathcal{AC}(X)$ has a unique final state without any outgoing transition.*

This variant for first occurrences satisfies properties analogous to Remark C.1:

Lemma C.3. *In $\mathcal{AC}(X)$, all transitions labeled with a letter that does not appear in any word of X go to the initial state. Moreover let u be a word without any factor in X except maybe as a suffix. Then the reading of u by $\mathcal{AC}(X)$ leads to the state labeled with the longest suffix of u that is also a prefix of a word of X .*

Proof. Let \mathcal{A} be the usual Aho-Corasick automaton on X . Then $\mathcal{AC}(X)$ (before the merge of all final states in f) is a subautomaton of \mathcal{A} , and therefore the first assertion is a direct consequence of Remark C.1. Let u be a word without any factor in X except maybe as a suffix. Then the path of the reading of u by \mathcal{A} does not visit any final state, except maybe the last state reached. Thus all this path is included in $\mathcal{AC}(X)$ and we conclude using Remark C.1. \square

A variant for a partition X_1, X_2 . When the set X is partitioned into two subsets X_1 and X_2 such that no word of X_1 (resp. X_2) is a factor of a word of X_2 (resp. X_1)⁸, we adapt the previous construction and build a deterministic automaton $\mathcal{AC}(X_1, X_2)$ which recognizes the same language as $\mathcal{AC}(X)$. But instead of merging all final states into a unique final state, we build two final states f_1 and f_2 corresponding to the first occurrence of a word of X_1 (resp. X_2). This construction is linear in time and space w.r.t. $\|X_1\| + \|X_2\|$. In what follows we will use this construction only when X_1 and X_2 are languages on disjoint alphabets, so that the factor independence condition is trivially satisfied.

Concatenation. Building an automaton $\mathcal{A}_1 \cdot \mathcal{A}_2$ recognizing 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 \mathcal{A}_1 has a unique final state without outgoing transitions. Indeed it is enough to merge the final state of \mathcal{A}_1 and the initial state of \mathcal{A}_2 into a unique state that is not initial (resp. not final), except when the initial state of \mathcal{A}_1 (resp. \mathcal{A}_2) is final. Note that the resulting automaton is deterministic and of size at most $|\mathcal{A}_1| + |\mathcal{A}_2|$, where the *size* $|\mathcal{A}|$ is the number of states of any automaton \mathcal{A} . This construction is done in constant time.

When the final state of \mathcal{A}_1 has no outgoing transitions except for a loop labeled by all letters of A and when \mathcal{L}_2 is of type $A^* \cdot \mathcal{L}$ for an arbitrary language \mathcal{L} , we can do the same construction to obtain an automaton recognizing the concatenation $\mathcal{L}_1 \cdot \mathcal{L}_2 = \mathcal{L}_1 \cdot A^* \cdot \mathcal{L}$. We just have to delete the loop on the final state of \mathcal{A}_1 before merging states.

In particular, according to this construction, the automaton obtained concatenating $\mathcal{AC}^\circ(X)$ and \mathcal{A} is $\mathcal{AC}(X) \cdot \mathcal{A}$. Therefore, even though $\mathcal{AC}(X)$ recognizes a language strictly included in A^*X , $\mathcal{AC}(X) \cdot \mathcal{A}$ recognizes $A^*XA^*\mathcal{L}$ when \mathcal{A} recognizes a language $A^*\mathcal{L}$.

Union. We say that an automaton is *almost complete* if for any letter a , all non final states have an outgoing transition labeled by a (notice that the only difference with complete automaton is that final states are allowed to miss some transitions). Let \mathcal{A}_1 and \mathcal{A}_2 be two deterministic automata that are almost complete⁹. We define the automaton $\mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$ as follows. We perform the Cartesian product of \mathcal{A}_1 and \mathcal{A}_2 beginning from the pair of initial states (see [16]). However we stop exploring a path when it enters a final state of \mathcal{A}_1 or \mathcal{A}_2 . Therefore in $\mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$ 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 $\mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$. Let \mathcal{L}_1 (resp. $\mathcal{L}_2, \mathcal{L}$) be the language recognized by \mathcal{A}_1 (resp. $\mathcal{A}_2, \mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$). Then \mathcal{L} is the set of words of $\mathcal{L}_1 \cup \mathcal{L}_2$ truncated after their first factor in $\mathcal{L}_1 \cup \mathcal{L}_2$. The language $(\mathcal{L}_1 \cup \mathcal{L}_2)A^* = \mathcal{L}A^*$ is recognized by the automaton $\mathcal{U}^\circ(\mathcal{A}_1, \mathcal{A}_2)$ with an additional loop labeled by all letters of

⁸This is a simple condition that allows us to define without ambiguity words ending with a first occurrence either in X_1 or in X_2 .

⁹ Notice that the automata $\mathcal{AC}(X)$, $\mathcal{AC}^\circ(X)$ and $\mathcal{AC}(X_1, X_2)$ satisfy these conditions.

A on the final state. Notice that $\mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$ (resp. $\mathcal{U}^\odot(\mathcal{A}_1, \mathcal{A}_2)$) is deterministic, almost complete (resp. complete) and has a unique final state without outgoing transitions (resp. whose only outgoing transition is a loop labeled by all letters of A). The complexity in time and space of these constructions is in $\mathcal{O}(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$.

C.2. Pin-permutation of size 1 and simple pin-permutations

Pin-permutation of size 1. When $\pi = 1$, we have seen in Subsection 4.3 that $\overleftarrow{\mathcal{L}}_\pi = A^* \mathcal{M}_2 A^*$ is recognized by the automaton \mathcal{A}_π of Figure 7 (p.21).

Simple pin-permutations. In this paragraph, for a simple permutation π whose set of pin words $P(\pi)$ is given, we build the automaton \mathcal{A}_π . The computation of $P(\pi)$ from π is discussed in Subsection 5.2 as a sub-procedure of the algorithm described in Section 5. The study that follows is based on the upcoming lemma.

Lemma C.4. *For every permutation π (not necessarily simple), we have*

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \mathcal{L}(u) = A^* \cdot E_\pi^S \cdot A^* \cup A^* \cdot \mathcal{M}_2 \cdot A^* \cdot E_\pi^{QS} \cdot A^*$$

where $E_\pi^S = \{\phi(u) \mid u \in P(\pi), u \text{ is strict}\}$ and $E_\pi^{QS} = \{\phi(u^{(2)}) \mid u = u^{(1)}u^{(2)} \in P(\pi), u \text{ is quasi-strict}\}$.

Proof. By definition of $\mathcal{L}(u)$ (see Definition 3.14 p.14),

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

Moreover, as can be seen on Figure 6 (p.10), for every quasi-strict pin word $u = u^{(1)}u^{(2)} \in P(\pi)$, the words $hu^{(2)}$ also belong to $P(\pi)$ for all $h \in \{1, 2, 3, 4\}$. This allows to write

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \mathcal{L}(u) = \left(\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict}}} A^* \phi(u) A^* \right) \bigcup \left(\bigcup_{h \in \{1, 2, 3, 4\}} A^* \phi(h) \cdot \bigcup_{\substack{u = u^{(1)}u^{(2)} \in P(\pi) \\ u \text{ quasi-strict}}} A^* \phi(u^{(2)}) A^* \right).$$

Hence

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \mathcal{L}(u) = A^* \cdot E_\pi^S \cdot A^* \cup A^* \cdot \mathcal{M}_2 \cdot A^* \cdot E_\pi^{QS} \cdot A^*,$$

concluding the proof. \square

Lemma C.5. *For every permutation π whose sets of strict and quasi-strict pin words (or equivalently E_π^S and E_π^{QS}) are given, one can build in time and space $\mathcal{O}(|E_\pi^S| \cdot |E_\pi^{QS}| \cdot |\pi|^2)$ a deterministic complete automaton \mathcal{A}_π^{SQS} having a unique final state without outgoing transitions except for a loop labeled by all letters of A that recognizes the language $\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}}(u)$.*

Proof. From Lemma C.4, we have

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}(u)} = \overleftarrow{A^* E_\pi^S A^*} \bigcup \overleftarrow{A^* \mathcal{M}_2 A^* E_\pi^{QS} A^*} = \left(A^* \overleftarrow{E_\pi^S} \bigcup (A^* \overleftarrow{E_\pi^{QS}} \cdot A^* \mathcal{M}_2) \right) A^*.$$

Recall that $\mathcal{AC}(\overleftarrow{E_\pi^S})$, $\mathcal{AC}(\overleftarrow{E_\pi^{QS}})$ and $\mathcal{AC}(\mathcal{M}_2)$ are automata recognizing respectively the set of words ending with a first occurrence of a word of $\overleftarrow{E_\pi^S}$, $\overleftarrow{E_\pi^{QS}}$ and \mathcal{M}_2 and obtained using the construction given in Subsection C.1. The sizes of the first two automata are respectively $\mathcal{O}(|E_\pi^S| \cdot |\pi|)$ and $\mathcal{O}(|E_\pi^{QS}| \cdot |\pi|)$, and the size of the third one is constant. Indeed for all w in E_π^S , $|w| = |\pi| + 1$ and for all w in E_π^{QS} , $|w| = |\pi|$ so that $\|E_\pi^S\| = |E_\pi^S| \cdot (|\pi| + 1)$ and $\|E_\pi^{QS}\| = |E_\pi^{QS}| \cdot |\pi|$.

Then the deterministic automaton \mathcal{A}_π^{SQS} is obtained as the union $\mathcal{U}^\cup(\mathcal{AC}(\overleftarrow{E_\pi^S}), \mathcal{AC}(\overleftarrow{E_\pi^{QS}}) \cdot \mathcal{AC}(\mathcal{M}_2))$ in time and space $\mathcal{O}(|E_\pi^S| \cdot |E_\pi^{QS}| \cdot |\pi|^2)$. \square

Lemma C.5 is used mostly in two special cases where *all* the pin words of π are strict or quasi-strict, and that we identify explicitly in the following remark.

Remark C.6. *By Theorem B.8 (p.38) the pin words encoding a simple permutation are either strict or quasi-strict and there are at most 48 of them. Therefore when π is a simple permutation, we take $\mathcal{A}_\pi = \mathcal{A}_\pi^{SQS}$ (from Lemma C.5) and the time and space complexity of the construction of \mathcal{A}_π is quadratic w.r.t. $|\pi|$, as soon as the pin words of π are given.*

Notice also that when $\pi = 12$, $\mathcal{A}_\pi = \mathcal{A}_\pi^{SQS}$ and the time and space complexity of the construction is $\mathcal{O}(1)$.

The above construction follows the general scheme announced at the beginning of the section, but it is not optimized. We actually can provide a more specific construction of \mathcal{A}_π whose complexity is *linear* when the permutation π is simple. This construction relies on the same idea as the one we give in [5].

Lemma C.7. *For every permutation π whose sets of strict and quasi-strict pin words (or equivalently E_π^S and E_π^{QS}) are given, one can build in time and space $\mathcal{O}((|E_\pi^S| + |E_\pi^{QS}|) \cdot |\pi|)$ a deterministic complete automaton \mathcal{A}_π^{SQS} having a unique final state without outgoing transitions except for a loop labeled by all letters of A that recognizes a language \mathcal{L}' such that*

$$\mathcal{L}' \cap \mathcal{M} = \bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}(u)} \cap \mathcal{M}.$$

Proof. Define $E_\pi = E_\pi^S \cup \mathcal{M}_2 \cdot E_\pi^{QS}$ and $\mathcal{L}' = A^* \cdot \overleftarrow{E_\pi} \cdot A^*$. Let us prove that

$$\mathcal{L}' \cap \mathcal{M} = \bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}(u)} \cap \mathcal{M}.$$

This will be enough to conclude the proof of Lemma C.7, setting $\mathcal{A}_\pi^{SQS} = \mathcal{AC}^\cup(\overleftarrow{E_\pi})$.

From Lemma C.4, we have

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \mathcal{L}(u) = A^* \cdot E_\pi^S \cdot A^* \cup A^* \cdot \mathcal{M}_2 \cdot A^* \cdot E_\pi^{\text{QS}} \cdot A^*.$$

Since $(A^* \cdot \mathcal{M}_2 \cdot A^* \cdot E_\pi^{\text{QS}} \cdot A^*) \cap \mathcal{M} = (A^* \cdot \mathcal{M}_2 \cdot E_\pi^{\text{QS}} \cdot A^*) \cap \mathcal{M}$ and $E_\pi = E_\pi^S \cup \mathcal{M}_2 \cdot E_\pi^{\text{QS}}$, we obtain

$$\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}(u)} \cap \mathcal{M} = A^* \cdot \overleftarrow{E_\pi} \cdot A^* \cap \mathcal{M},$$

concluding the proof. \square

Remark C.8. When π is a simple permutation, the automaton $\mathcal{A}_\pi^{\text{SQS}}$ of Lemma C.7 recognizes a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$. In the optimized alternative construction of \mathcal{A}_π mentioned in Subsection 4.3 and at the beginning of Appendix C, for a simple permutation π we take $\mathcal{A}_\pi = \mathcal{A}_\pi^{\text{SQS}}$ from Lemma C.7 and the time and space complexity of building the automaton \mathcal{A}_π is linear w.r.t. $|\pi|$ as soon as $P(\pi)$ is given.

C.3. Pin-permutations with a linear root: non-recursive case

W.l.o.g. (see Remark B.1 p.34), the only non-recursive case with a linear root is the one where $\pi = \oplus[\xi_1, \dots, \xi_r]$, every ξ_i being an increasing oscillation. Theorem B.12 (p.40) gives an explicit description of the elements of $P(\pi)$. These words are the concatenation of a pin word belonging to some $P(\oplus[\xi_i, \xi_{i+1}])$ with a pin word belonging to the shuffle product

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

To shorten the notations in the following, let us define

$$\chi_j^{(h)} = \overleftarrow{\phi(P^{(h)}(\xi_j))} \text{ for } h = 1 \text{ or } 3 \text{ and } 1 \leq j \leq r.$$

From Lemma B.13 (p.40), for all j , the pin words of $P^{(1)}(\xi_j)$ and $P^{(3)}(\xi_j)$ respectively are strict. Hence, the decomposition of $u \in P(\pi)$ into strong numeralled factors that is needed to describe $\overleftarrow{\mathcal{L}_\pi}$ (see p.19) is easily obtained and gives:

$$\overleftarrow{\mathcal{L}_\pi} = \bigcup_{1 \leq i \leq r-1} \left(\left(A^* \chi_1^{(1)}, \dots, A^* \chi_{i-1}^{(1)} \right) \sqcup \left(A^* \chi_r^{(3)}, \dots, A^* \chi_{i+2}^{(3)} \right) \right) \cdot \overleftarrow{\mathcal{L}_{\oplus[\xi_i, \xi_{i+1}]}}$$

In the above equation, we have $\overleftarrow{\mathcal{L}_{\oplus[\xi_i, \xi_{i+1}]}}$ where we might have expected to find $A^* \overleftarrow{\phi(P(\oplus[\xi_i, \xi_{i+1}]))} A^*$. The reason is that the term $A^* \overleftarrow{\phi(P(\oplus[\xi_i, \xi_{i+1}]))} A^*$ is not properly defined, since $P(\oplus[\xi_i, \xi_{i+1}])$ contains pin words that are not strict.

The automaton \mathcal{A}_π is then built by assembling smaller automata, whose construction is explained below.

Construction of $\mathcal{A}(\xi_i, \xi_j)$. Since for all i, j , the languages $\chi_i^{(1)}$ and $\chi_j^{(3)}$ – that contain at most two words – are defined on disjoint alphabets (see Remark B.14 p.40), we can use the construction given in Subsection C.1 to build the deterministic automata $\mathcal{A}(\xi_i, \xi_j) = \mathcal{AC}(\chi_i^{(1)}, \chi_j^{(3)})$. Figure 21 shows a diagram of $\mathcal{A}(\xi_i, \xi_j)$ and defines states s_{ij} , $f_{ij}^{(1)}$ and $f_{ij}^{(3)}$.

Lemma C.9. *For all i, j , the complexity in time and space of the construction of $\mathcal{A}(\xi_i, \xi_j)$ is $\mathcal{O}(|\xi_i| + |\xi_j|)$.*

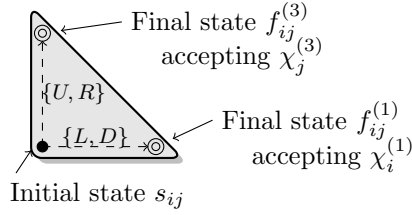


Figure 21: Atomic automaton $\mathcal{A}(\xi_i, \xi_j)$ used in the construction of \mathcal{A}_π .

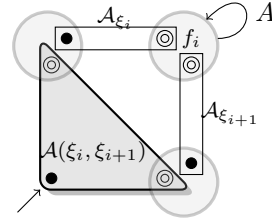


Figure 22: Automaton $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$.

Construction of \mathcal{A}_{ξ_i} . By Lemma B.15 (p.40), for any i , $|P(\xi_i)| \leq 48$, the pin words in $P(\xi_i)$ are explicit and all of them are either strict or quasi-strict except when $|\xi_i| = 3$.

If $|\xi_i| \neq 3$, from Lemma C.5 it is possible to build the deterministic automaton \mathcal{A}_{ξ_i} in quadratic time and space w.r.t. $|\xi_i|$, and from Lemma C.7 the construction can be optimized to be linear in time and space w.r.t. $|\xi_i|$.

If $|\xi_i| = 3$, $P(\xi_i)$ can be partitioned into two parts $P_{\text{sq}} \uplus P(12) \cdot h$ where $h = 4$ (resp. 2) if $\xi_i = 231$ (resp. 312) and P_{sq} is the set of strict and quasi-strict pin words of $P(\xi_i)$. With Lemma C.5 or C.7 we build the automaton $\mathcal{A}_{\xi_i}^{\text{sq}}$ corresponding to P_{sq} , and the automaton corresponding to $P(12) \cdot h$ is the concatenation of two basic automata, $\mathcal{AC}(\overleftarrow{\phi(h)})$ (where $\phi(h)$ for $h = 2$ or 4 is given p.12) and \mathcal{A}_{12} (see Remark C.6 p.57). Finally the automaton \mathcal{A}_{ξ_i} is the union $\mathcal{U}^\odot(\mathcal{A}_{\xi_i}^{\text{sq}}, \mathcal{AC}(\overleftarrow{\phi(h)}) \cdot \mathcal{A}_{12})$. As $|\xi_i| = 3$, \mathcal{A}_{ξ_i} is built in constant time.

Lemma C.10. *For all i , building \mathcal{A}_{ξ_i} is done in time and space $\mathcal{O}(|\xi_i|^2)$ with the classical construction and $\mathcal{O}(|\xi_i|)$ in the optimized version.*

Construction of $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$. We now explain how to build a deterministic automaton $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ recognizing $\overleftarrow{\mathcal{L}_{\oplus[\xi_i, \xi_{i+1}]}}$. Lemma B.18 (p.42) describes the pin words of $P(\oplus[\xi_i, \xi_{i+1}])$, proving the correctness of the following constructions.

If $|\xi_i| > 1$ and $|\xi_{i+1}| > 1$, we obtain $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ by gluing together automata $\mathcal{A}(\xi_i, \xi_{i+1})$, \mathcal{A}_{ξ_i} and $\mathcal{A}_{\xi_{i+1}}$ as shown in Figure 22. More precisely $f_{i(i+1)}^{(1)}$ (resp.

$f_{i(i+1)}^{(3)}$) and the initial state of $\mathcal{A}_{\xi_{i+1}}$ (resp. \mathcal{A}_{ξ_i}) are merged into a unique state that is neither initial nor final. The final states of \mathcal{A}_{ξ_i} and $\mathcal{A}_{\xi_{i+1}}$ are also merged into a unique final state f_i having a loop labeled by all letters of A .

If $|\xi_i| = 1$ and $|\xi_{i+1}| = 1$ then $\mathcal{A}^\oplus(\xi_i, \xi_{i+1}) = \mathcal{A}_{12}$ is built using Remark C.6.

Otherwise assume w.l.o.g. that $|\xi_i| = 1$ and $|\xi_{i+1}| > 1$. The set of pin words $P(\oplus[\xi_i, \xi_{i+1}])$ can be partitioned into two parts P_{sq} and P' , P_{sq} being the set of strict and quasi-strict pin words. If $|\xi_{i+1}| \neq 3$, then $P' = P(\xi_{i+1}) \cdot 3$ and $\mathcal{A}^\oplus(\xi_i, \xi_{i+1}) = \mathcal{U}^\odot \left(\mathcal{A}_{\oplus[\xi_i, \xi_{i+1}]}^{\text{sq}}, \mathcal{AC}(\overleftarrow{\phi(3)}) \cdot \mathcal{A}_{\xi_{i+1}} \right)$. If $|\xi_{i+1}| = 3$, then $P' = P(\xi_{i+1}) \cdot 3 \cup P(12) \cdot 3X$ where X is a direction, and we use again concatenation and union but performed on automata of constant size.

Note that in all cases $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ has a unique final state – that we denote f_i – without outgoing transitions except for the loop labeled by all letters of A .

Lemma C.11. *For all i , building the automaton $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ is done in time and space $\mathcal{O}(|\xi_i|^4 + |\xi_{i+1}|^4)$ with the classical construction and $\mathcal{O}(|\xi_i|^2 + |\xi_{i+1}|^2)$ in the optimized version.*

Proof. The complexity of the construction of $\mathcal{A}(\xi_i, \xi_{i+1})$ is linear w.r.t. $|\xi_i| + |\xi_{i+1}|$ from Lemma C.9 and the one of \mathcal{A}_{ξ_i} is quadratic – or linear in the optimized version – w.r.t. $|\xi_i|$ (see Lemma C.10). This concludes the proof except when $|\xi_i| = 1$ and $|\xi_{i+1}| \neq 1$ (or conversely). When $|\xi_i| = 1$, from Lemma B.18 (p.42), we have an explicit description of $P(\oplus[\xi_i, \xi_{i+1}])$ and $|P(\oplus[\xi_i, \xi_{i+1}])| \leq 192$. Hence, the complexity of the construction of $\mathcal{A}_{\oplus[\xi_i, \xi_{i+1}]}^{\text{sq}}$ when $|\xi_i| = 1$ is quadratic – or linear in the optimized version – w.r.t. $|\xi_{i+1}|$, using Lemmas C.5 and C.7. Then the result follows from the complexity of the union of two automata, which is proportional to the product of the sizes of the automata. \square

Assembling \mathcal{A}_π . According to the description of $\overleftarrow{\mathcal{L}}_\pi$ given p.58, the automata $\mathcal{A}(\xi_i, \xi_j)$ and $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ can be glued together to finish the construction of \mathcal{A}_π , as shown in Figure 23. More precisely for any i, j such that $1 \leq i < j \leq r$

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

and the final states f_i of the automata $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ are merged into a unique final state f having a loop labeled by all letters of A . The states q_{ij} defined above correspond to the shuffle product construction. To be more precise, taking the final state to be the merged state q_{ij} and adding a loop labeled by all letters of A on it, the accepted language would be $(A^* \chi_1^{(1)}, \dots, A^* \chi_{i-1}^{(1)}) \sqcup (A^* \chi_r^{(3)}, \dots, A^* \chi_{j+1}^{(3)}) \cdot A^*$ as it will be proved in the following. The automata $\mathcal{A}^\oplus(\xi_i, \xi_{i+1})$ in the second item above correspond to the concatenation with $\overleftarrow{\mathcal{L}}_{\oplus[\xi_i, \xi_{i+1}]}$.

Note that if $r = 2$, \mathcal{A}_π is $\mathcal{A}^\oplus(\xi_1, \xi_2)$.

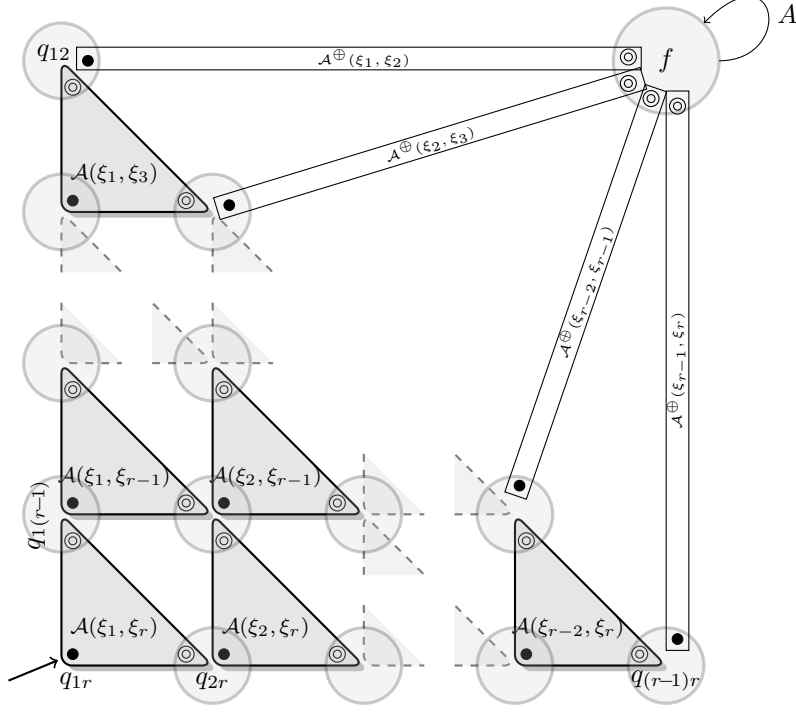


Figure 23: Automaton \mathcal{A}_π for $\pi = \oplus[\xi_1, \dots, \xi_r]$ where every ξ_i is an increasing oscillation.

Lemma C.12. *For any permutation π such that $\pi = \oplus[\xi_1, \dots, \xi_r]$, every ξ_i being an increasing oscillation, the construction of the automaton \mathcal{A}_π is done in time and space $\mathcal{O}(|\pi|^4)$ with the classical construction and $\mathcal{O}(|\pi|^2)$ in the optimized version.*

Proof. Denote by n the size of π , then $n = \sum_{i=1}^r |\xi_i|$. By construction, taking into account the merge of states:

$$|\mathcal{A}_\pi| \leq \sum_{i=1}^{r-2} \sum_{j=i+2}^r |\mathcal{A}(\xi_i, \xi_j)| + \sum_{i=1}^{r-1} |\mathcal{A}^\oplus(\xi_i, \xi_{i+1})|$$

and from Lemmas C.9 and C.11, in the classical construction

$$|\mathcal{A}_\pi| = \mathcal{O} \left(\sum_{i=1}^{r-2} \sum_{j=i+2}^r (|\xi_i| + |\xi_j|) + \sum_{i=1}^{r-1} (|\xi_i|^4 + |\xi_{i+1}|^4) \right) = \mathcal{O}(n^4)$$

and in the optimized version

$$|\mathcal{A}_\pi| = \mathcal{O} \left(\sum_{i=1}^{r-2} \sum_{j=i+2}^r (|\xi_i| + |\xi_j|) + \sum_{i=1}^{r-1} (|\xi_i|^2 + |\xi_{i+1}|^2) \right) = \mathcal{O}(n^2).$$

We conclude the proof noticing that all these automata are built in linear time w.r.t. their size. \square

Correctness of the construction. We now prove that the automaton \mathcal{A}_π given in Figure 23 recognizes the language $\tilde{\mathcal{L}}_\pi$, for $\pi = \oplus[\xi_1, \dots, \xi_r]$, every ξ_i being an increasing oscillation.

Definition C.13. Let \mathcal{A} be a deterministic complete automaton over the alphabet A whose set of states is Q and let u be a word in A^* . We define $\text{trace}_{\mathcal{A}}(u)$ as the word of $Q^{|u|+1}$ that consists in the states of the automaton that are visited when reading u from the initial state of \mathcal{A} , and for all $q \in Q$ we define $q \cdot u$ to be the state of \mathcal{A} reached when reading u from q .

Let u be a word in A^* . We define two parameters on u :

$$i(u) = \max\{i \in \{0, r\} \mid u \in A^* \cdot \chi_1^{(1)} \cdot A^* \cdot \chi_2^{(1)} \dots A^* \cdot \chi_i^{(1)} \cdot A^*\}, \text{ and}$$

$$j(u) = \min\{j \in \{1, r+1\} \mid u \in A^* \cdot \chi_r^{(3)} \cdot A^* \cdot \chi_{r-1}^{(3)} \dots A^* \cdot \chi_j^{(3)} \cdot A^*\}.$$

Remark C.14. Every word u such that $i(u) \geq i$ and $j(u) \leq j$ belongs to

$$\left((A^* \chi_1^{(1)}, A^* \chi_2^{(1)}, \dots, A^* \chi_i^{(1)}) \sqcup (A^* \chi_r^{(3)}, A^* \chi_{r-1}^{(3)}, \dots, A^* \chi_j^{(3)}) \right) \cdot A^*.$$

Proof. By definition, u belongs to $A^* \cdot \chi_1^{(1)} \cdot A^* \cdot \chi_2^{(1)} \dots A^* \cdot \chi_i^{(1)} \cdot A^*$ and to $A^* \cdot \chi_r^{(3)} \cdot A^* \cdot \chi_{r-1}^{(3)} \dots A^* \cdot \chi_j^{(3)} \cdot A^*$. Moreover, by Remark B.14 (p.40), all $\chi_k^{(1)}$ are words on the alphabet $\{L, D\}$ while all $\chi_k^{(3)}$ are words on $\{U, R\}$. These alphabets being disjoint, we conclude that u belongs to $\left((A^* \chi_1^{(1)}, A^* \chi_2^{(1)}, \dots, A^* \chi_i^{(1)}) \sqcup (A^* \chi_r^{(3)}, A^* \chi_{r-1}^{(3)}, \dots, A^* \chi_j^{(3)}) \right) \cdot A^*$. \square

The following lemma characterizes the first visit of state q_{ij} in \mathcal{A}_π :

Lemma C.15. Let Q be the set of states of \mathcal{A}_π (see Figure 23), $(i, j) \neq (1, r)$ and $u \in A^*$. Then $\text{trace}_{\mathcal{A}_\pi}(u) \in (Q \setminus q_{ij})^* \cdot q_{ij}$ if and only if $u = vw$ with either $w \in \chi_{i-1}^{(1)}$, $i(v) = i - 2$ and $j(v) = j + 1$; or $w \in \chi_{j+1}^{(3)}$, $i(v) = i - 1$ and $j(v) = j + 2$.

Proof. By induction on $r - j + i - 1$, using $\mathcal{A}(\xi_i, \xi_j) = \mathcal{AC}(\chi_i^{(1)}, \chi_j^{(3)})$. Notice that $r - j + i - 1$ is the number of automata $\mathcal{A}(\xi_k, \xi_\ell)$ that we need to go through before reaching q_{ij} . \square

Theorem C.16. *If $\pi = \oplus[\xi_1, \dots, \xi_r]$ where every ξ_i is an increasing oscillation then automaton \mathcal{A}_π given in Figure 23 recognizes the language $\overleftarrow{\mathcal{L}}_\pi$.*

Proof. Assume that $r \geq 3$ (otherwise $r = 2$, \mathcal{A}_π is $\mathcal{A}^\oplus(\xi_1, \xi_2)$ and the result trivially holds). We first prove that every word recognized by \mathcal{A}_π is in $\overleftarrow{\mathcal{L}}_\pi$. Let u be a word recognized by \mathcal{A}_π . Then $\text{trace}_{\mathcal{A}_\pi}(u)$ ends with the final state f of \mathcal{A}_π . As f is accessible only from some $\mathcal{A}^\oplus(\xi_k, \xi_{k+1})$, $\text{trace}_{\mathcal{A}_\pi}(u)$ contains some $q_{k(k+1)}$. Moreover for all $(i, j) \neq (1, r)$, every path from the initial state q_{1r} to q_{ij} contains $q_{(i-1)j}$ or $q_{i(j+1)}$. Therefore $\text{trace}_{\mathcal{A}_\pi}(u) \in q_{i_1 j_1} Q^* q_{i_2 j_2} Q^* \dots q_{i_{r-1} j_{r-1}} Q^* f$ where $(i_1, j_1) = (1, r)$, $(i_{r-1}, j_{r-1}) = (k, k+1)$ and for all ℓ , $(i_{\ell+1}, j_{\ell+1}) \in \{(i_\ell + 1, j_\ell), (i_\ell, j_\ell - 1)\}$. Hence by definition of $\mathcal{A}(\xi_i, \xi_j)$ and $\mathcal{A}^\oplus(\xi_k, \xi_{k+1})$ and by the expression of $\overleftarrow{\mathcal{L}}_\pi$ given p.58, $u \in \overleftarrow{\mathcal{L}}_\pi$.

Conversely, let $u \in \overleftarrow{\mathcal{L}}_\pi$. We want to prove that $q_{1r} \cdot u = f$, q_{1r} being the initial state of \mathcal{A}_π and f its final state. The expression of $\overleftarrow{\mathcal{L}}_\pi$ given p.58 ensures that there exists k such that $u = vw$ with $i(v) \geq k-1$, $j(v) \leq k+2$ and $w \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]}$. Let $u = v'w'$ with $v' = v_1 \dots v_s$ the shortest prefix of u such that $j(v') - i(v') \leq 3$, and set $i = i(v')$. Since v' is of minimal length, $j(v') = i + 3$ and there exists $v'' \in A^*$ such that $v = v'v''$. So $w' = v''w$ belongs to $A^* \cdot \overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]} = \overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]}$. Thus, using also Remark C.14, we have:

$$\begin{aligned} u &= \overbrace{\quad \quad \quad}^v \quad \quad \quad \overbrace{\quad \quad \quad}^w \\ &\quad \in (A^* \chi_1^{(1)}, \dots, A^* \chi_{k-1}^{(1)}) \sqcup (A^* \chi_r^{(3)}, \dots, A^* \chi_{k+2}^{(3)}) \quad \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]} \\ &= \overbrace{\quad \quad \quad}^{v'} \quad \quad \quad \overbrace{\quad \quad \quad}^{w'} \\ &\quad \in (A^* \chi_1^{(1)}, \dots, A^* \chi_i^{(1)}) \sqcup (A^* \chi_r^{(3)}, \dots, A^* \chi_{i+3}^{(3)}) \end{aligned}$$

Since v' is of minimal length, $i(v_1 \dots v_{s-1}) < i(v')$ or $j(v_1 \dots v_{s-1}) > j(v')$. Thus $v' = \bar{v}\bar{w}$ with either $\bar{w} \in \chi_{i(v')}^{(1)}$, $i(\bar{v}) = i(v') - 1$ and $j(\bar{v}) = j(v')$; or $\bar{w} \in \chi_{j(v')}^{(3)}$, $i(\bar{v}) = i(v')$ and $j(\bar{v}) = j(v') + 1$. By Lemma C.15, $\text{trace}_{\mathcal{A}_\pi}(v')$ ends with $q_{i(v')+1, j(v')-1}$.

Therefore $u = v'w'$ with $q_{1r} \cdot v' = q_{i+1, i+2}$ and $w' \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]}$.

If $i = k-1$ then $q_{1r} \cdot u = (q_{1r} \cdot v') \cdot w' = q_{k, k+1} \cdot w' = f$ as w' belongs to the language $\overleftarrow{\mathcal{L}}_{\oplus[\xi_k, \xi_{k+1}]}$ recognized by the automaton $\mathcal{A}^\oplus(\xi_k, \xi_{k+1})$.

If $i \leq k-3$. By definition $i(u) \geq k-1$ and $i(v') = i$, and as $u = v'w'$, $w' \in A^* \cdot \chi_{i+1}^{(1)} \cdot A^* \cdot \chi_{i+2}^{(1)} \cdot A^* \dots A^* \cdot \chi_{k-1}^{(1)} \cdot A^*$. Therefore as $i \leq k-3$, $w' \in A^* \cdot \chi_{i+1}^{(1)} \cdot A^* \cdot \chi_{i+2}^{(1)} \cdot A^*$ and w' belongs to the language $\overleftarrow{\mathcal{L}}_{\oplus[\xi_{i+1}, \xi_{i+2}]}$ recognized by the automaton $\mathcal{A}^\oplus(\xi_{i+1}, \xi_{i+2})$. Finally as $u = v'w'$ and $\text{trace}_{\mathcal{A}_\pi}(v')$ ends with $q_{i+1, i+2}$, $\text{trace}_{\mathcal{A}_\pi}(u)$ ends with f .

If $i \geq k+1$ then $j(v') \geq k+4$ and by symmetry of $i(u)$ and $j(u)$ the proof is similar to the previous case.

If $i = k-2$, as $v = v'v''$ and $i(v) \geq k-1$ and $i(v') = i$ then $v'' \in A^* \cdot \chi_{i+1}^{(1)} \cdot A^*$. Moreover $w \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_{i+2}, \xi_{i+3}]}$ so $w' = v''w \in A^* \cdot \chi_{i+1}^{(1)} \cdot \overleftarrow{\mathcal{L}}_{\oplus[\xi_{i+2}, \xi_{i+3}]}$. Therefore $w' \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_{i+1}, \xi_{i+2}, \xi_{i+3}]}$ and by Theorem 3.15 (p.14), $w' \in \overleftarrow{\mathcal{L}}_{\oplus[\xi_{i+1}, \xi_{i+2}]}$. Hence w' is recognized by $\mathcal{A}^\oplus(\xi_{i+1}, \xi_{i+2})$ so $q_{1r} \cdot u = f$ (since $q_{1r} \cdot v' = q_{i+1, i+2}$).

If $i = k$, by symmetry of $i(u)$ and $j(u)$ the proof is similar to the previous case, concluding the proof of Theorem C.16. \square

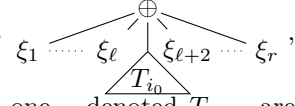
Remark C.17. With the optimized construction of \mathcal{A}_π , we prove similarly that \mathcal{A}_π recognizes a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$.

We end this subsection with a remark which will be useful in Subsection C.6.

Remark C.18. Let $\pi^{(1)} = \oplus[\xi_2, \dots, \xi_r]$ be the pattern of π obtained by deletion of the elements of ξ_1 . If $r \geq 3$ then $\mathcal{A}_{\pi^{(1)}}$ is obtained by taking q_{2r} (see Figure 23) as initial state and by considering only the states of \mathcal{A}_π that are accessible from q_{2r} . Thus in \mathcal{A}_π the language recognized from q_{2r} is $\overleftarrow{\mathcal{L}}_{\pi^{(1)}}$. If $r = 2$ then $\pi^{(1)} = \xi_2$, $\mathcal{A}_{\pi^{(1)}}$ is also a part of $\mathcal{A}^\oplus(\xi_1, \xi_2) = \mathcal{A}_\pi$ and $\overleftarrow{\mathcal{L}}_{\pi^{(1)}}$ is the language recognized from the bottom right state of Figure 22 (p.59). The same property holds with the pattern $\pi^{(r)} = \oplus[\xi_1, \dots, \xi_{r-1}]$, the state $q_{1(r-1)}$ and the top left state of Figure 22.

C.4. Pin-permutations with a linear root: recursive case

Suppose w.l.o.g. that the decomposition tree of π is



i.e., the root has label \oplus and all of its children but one – denoted T_{i_0} – are increasing oscillations. Then the automaton $\mathcal{A}(T_{i_0}) = \mathcal{A}_\rho$ associated to the permutation ρ whose decomposition tree is T_{i_0} is recursively obtained. We explain how to build \mathcal{A}_π from \mathcal{A}_ρ .

If $\pi \notin \mathcal{H}$, i.e. if π does not satisfy any condition of Figure 17 (p.44). Then Theorem B.19 (p.43) ensures that $P(\pi) = P(\rho) \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$ with

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

This characterization translates into the following expression for $\overleftarrow{\mathcal{L}}_\pi$:

$$\overleftarrow{\mathcal{L}}_\pi = \left(\left(A^\star \chi_1^{(1)}, \dots, A^\star \chi_\ell^{(1)} \right) \sqcup \left(A^\star \chi_r^{(3)}, \dots, A^\star \chi_{\ell+2}^{(3)} \right) \right) \cdot \overleftarrow{\mathcal{L}}_\rho$$

with the notations $\chi_j^{(h)}$ from p.58.

To deal with the shuffle product, we use again the automata $\mathcal{A}(\xi_i, \xi_j)$ whose initial and final states are $s_{ij}, f_{ij}^{(1)}$ and $f_{ij}^{(3)}$ (see Figure 21 p.59). We furthermore introduce the deterministic automata $\mathcal{A}^{(1)}(\xi_i) = \mathcal{AC}(\chi_i^{(1)})$ for $1 \leq i \leq \ell$ and $\mathcal{A}^{(3)}(\xi_j) = \mathcal{AC}(\chi_j^{(3)})$ for $\ell + 2 \leq j \leq r$ whose initial and final states are denoted respectively $s_i^{(1)}, f_i^{(1)}, s_j^{(3)}$ and $f_j^{(3)}$. The automaton $\mathcal{A}^{(1)}(\xi_i)$ (resp. $\mathcal{A}^{(3)}(\xi_j)$) corresponds to the reading of parts of $\mathfrak{P}_{(\ell)}^{(1)}$ (resp. $\mathfrak{P}_{(\ell+2)}^{(3)}$) in the shuffle product $\mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$ after all the parts of $\mathfrak{P}_{(\ell+2)}^{(3)}$ (resp. $\mathfrak{P}_{(\ell)}^{(1)}$) are read.

With these notations, the language $\overleftarrow{\mathcal{L}}_\pi$ associated to π is the one recognized by the automaton \mathcal{A}_π given in Figure 24 where the following merges are performed:

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

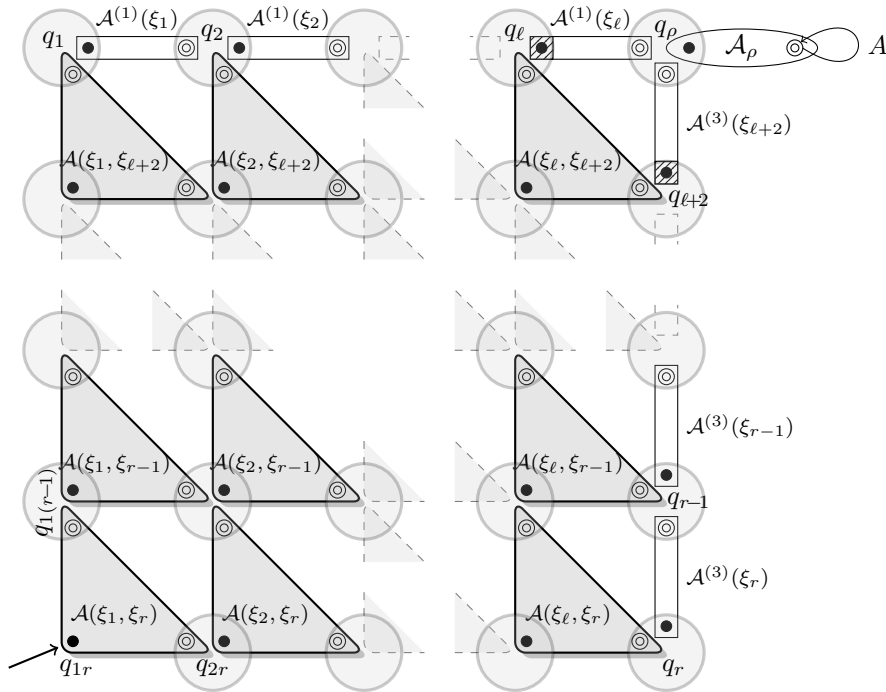


Figure 24: The automaton \mathcal{A}_π for $\pi = \oplus[\xi_1, \dots, \xi_\ell, \rho, \xi_{\ell+2}, \dots, \xi_r]$, where every ξ_i but ρ is an increasing oscillation (in the case $\pi \notin \mathcal{H}$).

Note that if $\ell = 0$ (resp. $r = \ell + 1$) i.e., if T_{i_0} is the first (resp. last) child, then only the automaton \mathcal{A}_ρ and the automata $\mathcal{A}^{(3)}(\xi_j)$ (resp. $\mathcal{A}^{(1)}(\xi_i)$) appear in \mathcal{A}_π whose initial state is then q_r (resp. q_1).

The proof that the automaton \mathcal{A}_π obtained by this construction recognizes $\overleftarrow{\mathcal{L}_\pi}$ is omitted. However this construction is very similar to the non-recursive case of the previous section where the proofs are detailed.

Lemma C.19. *For any pin-permutation $\pi = \oplus[\xi_1, \dots, \xi_\ell, \rho, \xi_{\ell+2}, \dots, \xi_r]$ such that every ξ_i but ρ is an increasing oscillation and $\pi \notin \mathcal{H}$, the construction of the automaton \mathcal{A}_π (see Figure 24) is done in time and space $\mathcal{O}((|\pi| - |\rho|)^2)$ plus the additional complexity due to the construction of \mathcal{A}_ρ , both in the classical and the optimized construction.*

Proof. First notice that $|\pi| - |\rho| = \sum_{i=1, i \neq \ell+1}^r |\xi_i|$. Moreover, taking into account the merge of states:

$$|\mathcal{A}_\pi| \leq \sum_{i=1}^{\ell} \sum_{j=\ell+2}^r |\mathcal{A}(\xi_i, \xi_j)| + \sum_{i=1}^{\ell} |\mathcal{A}^{(1)}(\xi_i)| + \sum_{j=\ell+2}^r |\mathcal{A}^{(3)}(\xi_j)| + |\mathcal{A}_\rho|.$$

From Lemma C.9 (p.59) and the fact that $|P^{(1)}(\xi_i)| \leq 2$ and $|P^{(3)}(\xi_j)| \leq 2$ (see Remark B.14 p.40), it follows that

$$|\mathcal{A}_\pi| - |\mathcal{A}_\rho| = \mathcal{O} \left(\sum_{i=1}^{\ell} \sum_{j=\ell+2}^r (|\xi_i| + |\xi_j|) + \sum_{\substack{i=1 \\ i \neq \ell+1}}^r |\xi_i| \right) = \mathcal{O}((|\pi| - |\rho|)^2),$$

concluding the proof, since the time of the construction is linear w.r.t. the size of the automaton. \square

We end this paragraph with a remark which will be useful in Subsection C.6.

Remark C.20. *If $\ell \neq 0$, let $\pi^{(1)}$ be the pattern of π obtained by deletion of the elements of ξ_1 . Then $\mathcal{A}_{\pi^{(1)}}$ is obtained by taking q_{2r} (see Figure 24) as initial state and by considering only the states of \mathcal{A}_π that are accessible from q_{2r} . Thus in \mathcal{A}_π the language recognized from q_{2r} is $\overleftarrow{\mathcal{L}_{\pi^{(1)}}}$. If $r \neq \ell+1$ the same property holds with the pattern $\pi^{(r)}$ (obtained by deletion of the elements of ξ_r) and the state $q_{1(r-1)}$. We take the convention that $q_{1(\ell+1)} = q_1$, $q_{(\ell+1)r} = q_r$ and $q_{(\ell+1)(\ell+1)} = q_\rho$.*

If $\pi \in \mathcal{H}$, i.e. if one of the conditions given in Figure 17 (p.44) holds for π . Then Theorem B.19 (p.43) ensures that $P(\pi)$ is the union of the set $P_0 = P(\rho) \cdot \mathfrak{P}_{(\ell)}^{(1)} \sqcup \mathfrak{P}_{(\ell+2)}^{(3)}$ that we consider in the previous paragraph and some other sets that are very similar, all ending with the same kind of shuffle product. As the automaton \mathcal{A}_π recognizes reversed words, these similar ends lead to similar beginnings in the automaton. So the automaton \mathcal{A}_π has the same general structure as the automaton \mathcal{A} of Figure 24 but some transitions are added to account for the words in $P(\pi)$ not belonging to P_0 .

More precisely \mathcal{A}_π is obtained performing the following modifications on the automaton \mathcal{A} of Figure 24. First we add new paths as depicted in the last

column of Figure 17 (p.44). These paths start in the shaded states q_ℓ or $q_{\ell+2}$ of Figure 24 and arrive in marked states of \mathcal{A}_ρ . If a path is labeled in Figure 17 by a word w with s letters we build $s - 1$ new states and s transitions such that the reading of w from the shaded state leads to the corresponding marked state of \mathcal{A}_ρ . These marked states may be seen as initial states of subautomata: in Figure 17, for all Y , q_Y is a state of \mathcal{A}_ρ such that the language recognized from q_Y is $\overleftarrow{\mathcal{L}_\sigma}$, where σ is the permutation whose diagram is Y . The way in which such states of \mathcal{A}_ρ are marked is explained in Subsection C.6.

Moreover to keep the resulting automaton deterministic and complete when adding these new paths we have to make some other changes. Notice that state q_ℓ (resp. $q_{\ell+2}$) is the initial state of $\mathcal{A}^{(1)}(\xi_\ell) = \mathcal{AC}(\overleftarrow{\chi_\ell^{(1)}})$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2}) = \mathcal{AC}(\overleftarrow{\chi_{\ell+2}^{(3)}})$). Remark B.14 (p.40) ensures that $\chi_\ell^{(1)} = \phi(P^{(1)}(\xi_\ell))$ (resp. $\chi_{\ell+2}^{(3)} = \phi(P^{(3)}(\xi_{\ell+2}))$) is defined on the alphabet $\{L, D\}$ (resp. $\{U, R\}$). Therefore, from Lemma C.3 (p.54), transitions labeled by U or R (resp. L or D) leaving q_ℓ (resp. $q_{\ell+2}$) are loops on the initial state q_ℓ (resp. $q_{\ell+2}$) of $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$). Hence, as can be seen on see Figure 17, the new transitions leaving shaded states are labeled by directions that correspond to loops in \mathcal{A} . So we just have to delete the loops and replace them by the new transitions in order to preserve the determinism of the automaton.

Now we make some other changes to preserve completeness and ensure that even though we have deleted loops on shaded states, all words that were recognized by the automaton \mathcal{A} are still recognized by the modified automaton \mathcal{A}_π . As q_ℓ (resp. $q_{\ell+2}$) is not reachable from $q_{\ell+2}$ (resp. q_ℓ) we can handle separately new states reachable from q_ℓ and new states reachable from $q_{\ell+2}$. Let q_0 be q_ℓ (resp. $q_{\ell+2}$). Like in the Aho-Corasick algorithm we label any new state q reachable from q_0 by the shortest word labeling a path from q_0 to q . So these labels begin with U or R (resp. L or D) (see Figure 17). Notice that the states in the part $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$) of \mathcal{A} are also labeled in such a way, but their labels differ from the ones of the new states since they contain only letters L or D (resp. U or R). By Lemma C.3 (p.54), we know that in $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$) all transitions labeled by U or R (resp. L or D) go to q_0 , therefore we replace them by transitions going to the new state labeled by U or R (resp. L or D) if such a new state exists (otherwise we keep the transition going to q_0). We complete the construction by adding missing transitions from the states newly created: for any such state q , the transition from q labeled by Z goes to the longest suffix of $q \cdot Z$ that is a state of the automaton – either a new state or a state of $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$).

The proof that the automaton \mathcal{A}_π obtained by this construction recognizes $\overleftarrow{\mathcal{L}_\pi}$ is omitted to avoid the examination of the eight cases of Figure 17. However, it is similar to the proof of Theorem C.23 (p.70), with some of the difficulties released (since labels on the new paths are explicit in Figure 17, while they are not in the proof of Theorem C.23).

Lemma C.21. *The complexity of building \mathcal{A}_π given in Lemma C.19 (p.66) still holds if $\pi \in \mathcal{H}$.*

Proof. When $\pi \in \mathcal{H}$, the construction of \mathcal{A}_π is the same as in the case $\pi \notin \mathcal{H}$, with some new paths added. There are at most four new paths, $\mathcal{O}(|\rho|)$ new states in each path, $\mathcal{O}(|\rho|)$ transitions from these new states, and the modification of transitions in $\mathcal{A}^{(1)}(\xi_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2})$) is done in $\mathcal{O}(|\xi_\ell|)$ (resp. $\mathcal{O}(|\xi_{\ell+2}|)$). So in the construction of \mathcal{A}_π described above, we have to add a time and space complexity $\mathcal{O}(|\rho| + |\xi_\ell| + |\xi_{\ell+2}|)$ w.r.t. the case $\pi \notin \mathcal{H}$. As $|\xi_\ell| + |\xi_{\ell+2}| = \mathcal{O}(|\pi| - |\rho|)$ and as the complexity of the construction of \mathcal{A}_ρ is bigger than $\mathcal{O}(|\rho|)$, this does not change the overall estimation of the complexity of the construction of \mathcal{A}_π given in Lemma C.19. \square

C.5. Pin-permutations with a prime root: recursive case

C.5.1. Exactly one child of the root is not a leaf.

Let $\pi = \bullet \cdots \bullet \begin{array}{c} \alpha \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \\ \triangle \\ T \end{array} \bullet \cdots \bullet$ where α is a simple permutation all of whose children but T are leaves. Denote by ρ the permutation whose decomposition tree is T , and by x the point of α expanded by T .

Recall that $Q_x(\alpha)$ (see Definition B.23 p.48) denotes the set of strict pin words obtained by deleting the first letter of quasi-strict pin words of α whose first point read in α is x .

If π does not satisfy condition (C) (see Definition B.21 p.47). Then from Theorem B.25 (p.49), $P(\pi) = P(\rho) \cdot Q_x(\alpha)$. So $\overleftarrow{\mathcal{L}}_\pi = A^\star \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}}_\rho$ and since $\overleftarrow{\mathcal{L}}_\rho = A^\star \cdot \overleftarrow{\mathcal{L}}_\rho$ the automaton \mathcal{A}_π recognizing $\overleftarrow{\mathcal{L}}_\pi$ is obtained by the concatenation of $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ with \mathcal{A}_ρ , which is recursively obtained.

If π satisfies condition (C) and $|T| \geq 3$. Then by Theorem B.25 (p.49) – and using the notations of this theorem, $P(\pi)$ contains $P(\rho) \cdot Q_x(\alpha)$ and some other words. Defining T' as in condition (C), ρ' the permutation whose decomposition tree is T' , and w the unique word encoding the unique reading of the remaining leaves in π after T' is read when T is read in two pieces, these other words are $P(\rho') \cdot w$. Note that from Lemma B.27 (p.50) w is a strict pin word. So $\overleftarrow{\mathcal{L}}_\pi = A^\star \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}}_\rho \cup A^\star \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}}_{\rho'}$. The skeleton of \mathcal{A}_π is the concatenation of the automaton $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ with \mathcal{A}_ρ and then as in the recursive case with a linear root, we add some new transitions to account for the words in $P(\rho') \cdot w$.

Denoting Z the last letter of w (i.e., the first letter of $\overleftarrow{\phi(w)}$), Lemma B.27 ensures that no word of $\overleftarrow{\phi(Q_x(\alpha))}$ contains Z and therefore by Lemma C.3 (p.54) all the transitions labeled by Z in the automaton $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ go to its initial state, denoted q_0 . We built an automaton \mathcal{A} by performing the following modifications on $\overleftarrow{\mathcal{AC}(\phi(Q_x(\alpha)))}$: remove the loop labeled by Z on q_0 and add a path reading $\overleftarrow{\phi(w)}$ from q_0 to a new final state f' . Label all states q of \mathcal{A} by the shortest word labeling a path from the initial state q_0 to q . Replace any transition labeled by Z from a state q of $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ to q_0 by a transition from q to the new state labeled by Z . Finally complete the automaton with

transitions from the states of the added path: for all such states q but f' , the transition from q labeled by a goes to the longest suffix of $q \cdot a$ that is a state of the automaton – either a new state or a pre-existing state. Notice that the automaton \mathcal{A} we obtain is almost complete and has exactly two final states, without outgoing transitions: f – the unique final state of $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ – and f' .

The automaton \mathcal{A}_π is then obtained from \mathcal{A} and \mathcal{A}_ρ by merging f with the initial state q_T of \mathcal{A}_ρ and f' with a marked state $q_{T'}$ (see Subsection C.6) of \mathcal{A}_ρ which is a state from which the recognized language is $\overleftarrow{\mathcal{L}_{\rho'}}$. This construction is shown in Figure 25.

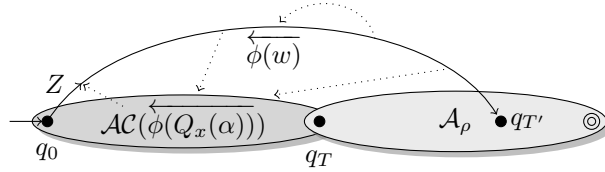


Figure 25: Automaton \mathcal{A}_π for $\pi = \alpha[1, \dots, 1, \rho, 1, \dots, 1]$.

Notice that the automaton \mathcal{A} obtained from $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ is somehow very similar to $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}, \{\overleftarrow{\phi(w)}\})$ but because $\overleftarrow{\phi(w)}$ has a suffix in $\overleftarrow{\phi(Q_x(\alpha))}$ (from Lemma B.27), the sets of words $X_1 = \overleftarrow{\phi(Q_x(\alpha))}$ and $X_2 = \{\overleftarrow{\phi(w)}\}$ do not satisfy the independence condition required in our construction of $\mathcal{AC}(X_1, X_2)$.

Lemma C.22. *The automaton \mathcal{A} of the above construction recognizes the set of words ending with a first occurrence of a word of $\overleftarrow{\phi(Q_x(\alpha))}$. Moreover for any word u recognized by \mathcal{A} , $q_0 \cdot u = f'$ if $\overleftarrow{\phi(w)}$ is a suffix of u , and $q_0 \cdot u = f$ otherwise.*

Proof. From Lemma B.27 (p.50), there exists a word $\bar{w} \in \overleftarrow{\phi(Q_x(\alpha))}$ and a letter $Z \in A$ such that $\overleftarrow{\phi(w)} = Z\bar{w}$. Moreover no word of $\overleftarrow{\phi(Q_x(\alpha))}$ contains Z .

Therefore by construction, merging states f and f' of \mathcal{A} into a unique final state, we would obtain the automaton $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))} \cup \{\overleftarrow{\phi(w)}\})$. Consequently, since $\overleftarrow{\phi(w)}$ has a suffix in $\overleftarrow{\phi(Q_x(\alpha))}$, the automaton \mathcal{A} recognizes the set of words ending with a first occurrence of a word of $\overleftarrow{\phi(Q_x(\alpha))}$.

Let u be a word ending with its first occurrence of a word of $\overleftarrow{\phi(Q_x(\alpha))}$, then u does not have any factor in $\overleftarrow{\phi(Q_x(\alpha))} \cup \{\overleftarrow{\phi(w)}\}$ except as a suffix. Lemma C.3 (p.54) ensures that $q_0 \cdot u$ is the state labeled with longest suffix of u that is also a prefix of a word of $\overleftarrow{\phi(Q_x(\alpha))} \cup \{\overleftarrow{\phi(w)}\}$, concluding the proof. \square

Lemma C.22 allows us to prove the correctness of the above construction of \mathcal{A}_π . The idea is the following: if $u \in A^* \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}_\rho}$ (resp. $A^* \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}}$) and if $\text{trace}_{\mathcal{A}_\pi}(u)$ contains $q_{T'}$ (resp. q_T) and not q_T (resp. $q_{T'}$) before, then u is still accepted by \mathcal{A}_π since $\overleftarrow{\mathcal{L}_\rho} \subseteq \overleftarrow{\mathcal{L}_{\rho'}}$ (resp. $\overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}} \subseteq \overleftarrow{\mathcal{L}_\rho}$). This is formalized in the following theorem.

Theorem C.23. *The automaton \mathcal{A}_π obtained by the above construction recognizes $\overleftarrow{\mathcal{L}}_\pi$.*

Proof. Recall that $\overleftarrow{\mathcal{L}}_\pi = A^* \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}}_\rho \cup A^* \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}}_{\rho'}$. The above construction ensures that every word accepted by \mathcal{A}_π belongs to the language $\overleftarrow{\mathcal{L}}_\pi$. Conversely let us prove that every word of $\overleftarrow{\mathcal{L}}_\pi$ is accepted by \mathcal{A}_π .

Let u be a word of $\overleftarrow{\mathcal{L}}_\pi$. From Lemma B.27 (p.50), there is a word $\bar{w} \in \overleftarrow{\phi(Q_x(\alpha))}$ and a letter $Z \in A$ such that $\overleftarrow{\phi(w)} = Z\bar{w}$. Therefore u has a factor in $\overleftarrow{\phi(Q_x(\alpha))}$. Hence we can decompose u uniquely as $u = u_1 u_2$ where u_1 is the prefix of u ending with the first occurrence of a factor in $\overleftarrow{\phi(Q_x(\alpha))}$. Consequently from Lemma C.22, $q_0 \cdot u_1$ is either q_T or $q_{T'}$, namely $q_0 \cdot u_1 = q_{T'}$ if $\overleftarrow{\phi(w)}$ is a suffix of u_1 and $q_0 \cdot u_1 = q_T$ otherwise.

Moreover, since u belongs to $\overleftarrow{\mathcal{L}}_\pi$, and because by definition $\overleftarrow{\mathcal{L}}_\rho = A^* \cdot \overleftarrow{\mathcal{L}}_\rho$ (and similarly for ρ'), we deduce that u_2 belongs to $\overleftarrow{\mathcal{L}}_\rho$ or $\overleftarrow{\mathcal{L}}_{\rho'}$. Let us finally notice that, since $\rho' \leq \rho$, Theorem 3.15 (p.14) ensures that $\overleftarrow{\mathcal{L}}_\rho \subseteq \overleftarrow{\mathcal{L}}_{\rho'}$ thus $u_2 \in \overleftarrow{\mathcal{L}}_{\rho'}$.

If $q_0 \cdot u_1 = q_{T'}$ then as $u_2 \in \overleftarrow{\mathcal{L}}_{\rho'}$, u is recognized by \mathcal{A}_π . Assume on the contrary that $q_0 \cdot u_1 = q_T$. Then $q_0 \cdot u = q_T \cdot u_2$ and by definition of q_T it is enough to prove that $u_2 \in \overleftarrow{\mathcal{L}}_\rho$.

Assume first that $u \notin A^* \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}}_{\rho'}$. Then since $u \in \overleftarrow{\mathcal{L}}_\pi$, we have $u \in A^* \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}}_\rho$. Because u_1 ends with the first occurrence of a factor of $\overleftarrow{\phi(Q_x(\alpha))}$, we deduce that $u_2 \in A^* \cdot \overleftarrow{\mathcal{L}}_\rho = \overleftarrow{\mathcal{L}}_\rho$.

Otherwise $u \in A^* \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}}_{\rho'}$. Recall that u_1 is the prefix of u ending with the first occurrence of a factor of $\overleftarrow{\phi(Q_x(\alpha))}$. First (using also Lemma C.22 and $q_0 \cdot u_1 = q_T$), this implies that $\overleftarrow{\phi(w)}$ is not a suffix of u_1 . And second, this also implies that $\overleftarrow{\phi(w)}$ is not a factor of u_1 . But by assumption $\overleftarrow{\phi(w)}$ is a factor of u . We claim that the first occurrence of $\overleftarrow{\phi(w)}$ in u starts after the end of u_1 . We have just proved that $\overleftarrow{\phi(w)}$ is not a factor of u_1 . Moreover, $\overleftarrow{\phi(w)} = Z\bar{w}$ starts with the letter Z , and from Lemma B.27 (p.50) the $|\bar{w}|$ last letters of u_1 are different from Z (recall that all words of $\overleftarrow{\phi(Q_x(\alpha))}$ have the same length $|\alpha| = |\bar{w}|$). This proves our claim, and consequently, $u_2 \in A^* \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}}_{\rho'}$. Let $v \in A^*$ and $v' \in \overleftarrow{\mathcal{L}}_{\rho'}$ such that $u_2 = v \cdot \overleftarrow{\phi(w)} \cdot v'$. From Lemma B.27 p.50, denoting by w' the suffix of length 2 of w , for all u' in $P(\rho')$, $u' \cdot \phi^{-1}(w')$ belongs to $P(\rho)$. Therefore $\overleftarrow{w'} \overleftarrow{\mathcal{L}}_{\rho'} \subseteq \overleftarrow{\mathcal{L}}_\rho$. But $v' \in \overleftarrow{\mathcal{L}}_{\rho'}$ and $\overleftarrow{w'}$ is a prefix of $\overleftarrow{\phi(w)}$, thus $u_2 = v \cdot \overleftarrow{\phi(w)} \cdot v' \in A^* \cdot \overleftarrow{w'} \cdot A^* \cdot \overleftarrow{\mathcal{L}}_{\rho'} \subseteq \overleftarrow{\mathcal{L}}_\rho$, concluding the proof. \square

Remark C.24. *With the optimized construction of \mathcal{A}_π , we prove similarly that \mathcal{A}_π recognize a language \mathcal{L}'_π such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}}_\pi \cap \mathcal{M}$.*

If π satisfies condition (C) and $|T| = 2$. Then the construction is no more recursive. Permutation π and its pin words are explicit. More precisely from

Theorem B.25 (p.49), $P(\pi) = P_{\{1,n\}}(\pi) \cup P_{\{2,n\}}(\pi) \cup P(T) \cdot Q_x(\alpha)$. Thus from Remark B.30 (p.51),

$$\overleftarrow{\mathcal{L}}_\pi = \left(\bigcup_{\substack{u \in P(\pi) \\ u \text{ strict or quasi-strict}}} \overleftarrow{\mathcal{L}}(u) \right) \cup A^\star \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}}_\rho.$$

Therefore \mathcal{A}_π is the automaton $\mathcal{U}^\odot(\mathcal{A}_\pi^{\text{sqs}}, \mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho)$.

Lemma C.25. *Let $\pi = \alpha[1, \dots, 1, \rho, 1, \dots, 1]$ where α is a simple permutation whose set $P(\alpha)$ of pin words is given. Then the construction of the automaton \mathcal{A}_π is done in time and space $\mathcal{O}(|\pi| - |\rho|)$ plus the additional time and space due to the construction of \mathcal{A}_ρ , except when π satisfies condition (C) and $|T| = 2$. In this latter case, the complexity is $\mathcal{O}(|\pi|^3)$ with the classical construction and $\mathcal{O}(|\pi|^2)$ in the optimized version.*

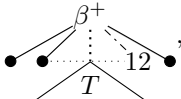
Proof. Recall that $Q_x(\alpha)$ contains words of length $|\alpha| - 1$. Its cardinality is smaller than the one of $P(\alpha)$, hence smaller than 48 (see Theorem B.8 p.38). Moreover $Q_x(\alpha)$ can be determined in linear time w.r.t. $|\alpha|$ as described in Remark B.26 (p.50). Consequently, $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ is built in time and space $\mathcal{O}(|\alpha|) = \mathcal{O}(|\pi| - |\rho|)$.

If π does not satisfy condition (C) then $\mathcal{A}_\pi = \mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho$, so that $|\mathcal{A}_\pi| = |\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})| + |\mathcal{A}_\rho|$ and the time complexity of this construction is $\mathcal{O}(|\pi| - |\rho|)$ plus the additional time to build \mathcal{A}_ρ .

If π satisfies condition (C) and $|T| \geq 3$, then $|w| = |\alpha|$ and by Remark B.29 (p.51), w is explicitly determined. Consequently, so is the additional path labeled by $\overleftarrow{\phi(w)}$ added to the automaton (see Figure 25). The modifications of the transitions between this path and $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ are performed in linear time w.r.t. the length of this path and $|\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})|$, i.e., in $\mathcal{O}(|\phi(w)| + |\alpha|) = \mathcal{O}(|\pi| - |\rho|)$. We conclude that \mathcal{A}_π is built in $\mathcal{O}(|\pi| - |\rho|)$ time and space plus the additional time and space to build \mathcal{A}_ρ .

If π satisfies condition (C) and $|T| = 2$, then $\mathcal{A}_\pi = \mathcal{U}^\odot(\mathcal{A}_\pi^{\text{sqs}}, \mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho)$. Recall that $P_{\text{sqs}}(\pi)$ is given in Remark B.30 (p.51) and contains 12 pin words. Hence, with the classical construction (resp. in the optimized version), from Lemma C.5 (p.56) (resp. Lemma C.7 p.57) and Remark B.30, we can build $\mathcal{A}_\pi^{\text{sqs}}$ in time and space $\mathcal{O}(|\pi|^2)$ (resp. $\mathcal{O}(|\pi|)$). Moreover since $|\rho| = 2$, \mathcal{A}_ρ is obtained in constant time, so that $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho$ is obtained in time and space $\mathcal{O}(|\pi| - |\rho|) = \mathcal{O}(|\pi|)$. Finally, \mathcal{A}_π is built in time and space $\mathcal{O}(|\pi|^3)$ (resp. $\mathcal{O}(|\pi|^2)$) with the classical (resp. optimized) construction. \square

C.5.2. Two children are not leaves.

Up to symmetry this means that $\pi =$ , where β^+ is an increasing quasi-oscillation, the permutation 12 expands an auxiliary point of β^+ and T expands the corresponding main substitution point of β^+ .

Theorem B.31 (p.52) ensures that the pin words encoding π are of the form $v.w$ where $v \in P(T)$ and w is a strict pin word of length $|\beta^+|$ uniquely determined by β^+ and the two points expanded in β^+ , and known explicitly from Remark B.32 (p.52).

Therefore $\overleftarrow{\mathcal{L}}_\pi = A^* \overleftarrow{\phi(w)} \overleftarrow{\mathcal{L}}_\rho$ where ρ is the permutation whose decomposition tree is T . The automaton \mathcal{A}_π recognizing $\overleftarrow{\mathcal{L}}_\pi$ is the concatenation of $\mathcal{AC}(\{\overleftarrow{\phi(w)}\})$ with \mathcal{A}_ρ , which is recursively obtained.

This construction is done in $\mathcal{O}(|\overleftarrow{\phi(w)}|) = \mathcal{O}(|\pi| - |\rho|)$ time and space in addition to the time and space complexity of the construction of \mathcal{A}_ρ .

C.6. Marking states

In our constructions of Subsections C.4 and C.5 we need transitions going to initial states of subautomata. We could duplicate the corresponding subautomata. But when these are recursively obtained an exponential blow-up can occur. To keep a polynomial complexity we replace duplication by the marking of these special states. The marking is made on the fly during the construction and we explain how in this subsection.

The need of creating a transition going to a marked state (of a subautomaton) happens only when building the automaton \mathcal{A}_π in Subsection C.4 for a permutation π whose decomposition tree has a linear root and satisfies a condition (iHj) of Figure 17 (p.44), or in Subsection C.5 for a permutation π whose decomposition tree has a prime root and satisfies condition (\mathcal{C}) (see Definition B.21 p.47) with $|T| \geq 3$.

In both cases we need to mark in the subautomaton \mathcal{A}_ρ with $\rho \leq \pi$ some states q_Y such that the language recognized taking q_Y as initial state is $\overleftarrow{\mathcal{L}}_\sigma$, where $\sigma \leq \rho$ is the permutation whose diagram (or decomposition tree) is Y .

As it appears in Figure 17 and in condition (\mathcal{C}), in almost all such situations, the marked state belongs to a subautomaton corresponding to a permutation ρ whose decomposition tree R has a linear root. There is only one situation where this root is prime: when π satisfies condition ($1H1+$). We first focus on this case.

Prime root. Let θ be a permutation of decomposition tree $R = \bullet \cdots \bullet \overset{\xi^+}{\triangle} \bullet \cdots \bullet$

where ξ^+ is an increasing oscillation, and let γ be the permutation whose decomposition tree is S . In the case where π satisfies condition ($1H1+$), we need to mark in the automaton \mathcal{A}_θ the state q such that when starting from q the language recognized is the one recognized by \mathcal{A}_γ . (Notice that w.r.t. the previous paragraph, we have changed the notations ρ to θ and σ to γ to avoid confusions with the notations used in Subsection C.5.)

The automaton \mathcal{A}_θ is obtained as described in Subsection C.5, 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}_θ is built and in particular on whether θ satisfies condition (\mathcal{C}) or not.

Recall that ξ^+ is an increasing oscillation. If ξ^+ has a size at least 5, it is not a quasi-oscillation, and θ does not satisfy condition (\mathcal{C}). Therefore \mathcal{A}_θ is

the concatenation of two automata the second of which is \mathcal{A}_γ whose initial state can be readily marked.

If ξ^+ has size 4, then $\xi^+ = 2413$ or 3142 is a quasi-oscillation and θ may satisfy condition (C). If it is not the case, \mathcal{A}_θ is obtained as above and so is the marking of state q . If on the contrary θ satisfies condition (C), the construction of \mathcal{A}_θ depends on $|S|$. If $|S| \geq 3$, \mathcal{A}_θ is again the concatenation of two automata the second one being \mathcal{A}_γ , but with some states and transitions added. As these transitions are not reachable from the initial state of \mathcal{A}_γ , we mark it as above. If $|S| = 2$, then R has size 5 and the construction is not recursive anymore. We want to mark in \mathcal{A}_θ a state q corresponding to the initial state of \mathcal{A}_γ . But in the construction of \mathcal{A}_θ in Subsection C.5, we have built an automaton \mathcal{A}' such that $\mathcal{A}_\theta = \mathcal{U}^\circ(\mathcal{A}_\theta^{\text{sq}}, \mathcal{A}' \cdot \mathcal{A}_\gamma)$. Therefore \mathcal{A}_θ is a Cartesian product and the state q has been replicated several times. As $|S| = 2$, \mathcal{A}_γ has a constant size, hence in this particular case we just duplicate it and mark its initial state instead of marking a state inside \mathcal{A}_θ .

Linear root. Consider now the case where the decomposition tree R of the permutation ρ has a linear root. The need of a marked state in \mathcal{A}_ρ happens only when the leftmost (resp. rightmost) child of R is a leaf z .

In almost all cases, the marked state q is such that the language accepted starting from q is the set of words encoding the readings of all nodes of R except the leaf z . There are at most two such leaves and from Remarks C.18 and C.20 (p.64 and 66), the corresponding marked states of \mathcal{A}_ρ (which is built as described in Subsection C.3 or C.4) are $q_{1(r-1)}$ and q_{2r} in Figure 23 (p.61) or 24 (p.65) – with ρ instead of π . There is however one exception, corresponding to the special case described in Remark C.18: when R has exactly two children, which are z and an increasing oscillations ξ . In this special case the construction of \mathcal{A}_ρ is not recursive anymore. Instead of marking in \mathcal{A}_ρ a state q corresponding to the initial state of \mathcal{A}_ξ , we just duplicate \mathcal{A}_ξ and mark its initial state. If $|\xi| < 4$ then $|\mathcal{A}_\xi| = \mathcal{O}(1)$. Otherwise ξ is a simple permutation and $|\mathcal{A}_\xi|$ is quadratic (or linear in the optimized complexity) w.r.t. $|\xi|$. In both cases $|\mathcal{A}_\xi| + |\mathcal{A}_\rho|$ has the same order as $|\mathcal{A}_\rho|$ and since the construction is not recursive, this does not change the overall complexity of the construction of \mathcal{A}_π .

The few cases where the marked state q is not as above (*i.e.*, is not such that the language accepted starting from q is the set of words encoding the readings of all nodes of R except z) correspond to state q_S of conditions (2H2 \star) and (1H2 \star) and states $q_{T \cup a}$ and $q_{T \cup b}$ of condition (2H3). In these cases, R has exactly two children: z and a subtree R' whose root is linear. Then the leftmost (resp. rightmost) child of R' is a leaf z' and the marked state q is such that the language accepted starting from q is the set of words encoding the readings of all nodes of R' except the leaf z' . We are in the same situation as above, so the states can be marked in the same way, except that now we have to mark states in $\mathcal{A}_{\rho'}$ instead of \mathcal{A}_ρ , where ρ' is the permutation whose decomposition tree is R' and $\mathcal{A}_{\rho'}$ is a subautomaton of \mathcal{A}_ρ built recursively.

Notice that we never create transitions going to marked states belonging to automata built more than two levels of recursion deeper. Indeed in all conditions

above the created transitions go to the automaton build in the previous step of recursion, except for conditions (2H3), (2H2★) and (1H2★) where two levels of recursion are involved.

C.7. Complexity analysis

Theorem C.26. *For every pin-permutation π of size n , \mathcal{A}_π is built in time and space $\mathcal{O}(n^2)$ in the optimized version and $\mathcal{O}(n^4)$ in the classical version.*

Proof. To build \mathcal{A}_π , we first need to decide which shape of Equation (★) is matched by the decomposition tree of π , and whether $\pi \in \mathcal{H}$ or whether π satisfies condition \mathcal{C} . The reader familiar with matching problems will be convinced that this can be done in $\mathcal{O}(n)$ time. In any case, a linear algorithm for this tree matching problem is detailed in Subsection 5.2 as a sub-procedure of the global algorithm of Section 5.

Then Theorem C.26 follows from the complexities of the previous constructions that are summarized in Table 2 in which we denote by ρ the permutation whose decomposition tree is T .

pin-permutation of size n	Complexity	Optimized	Lemma
size 1	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
simple	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	C.6, C.8
root \oplus non-recursive	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	C.12
root \oplus recursive, one child T is not an increasing oscillation	$\mathcal{O}((n - \rho)^2) +$ contribution for \mathcal{A}_ρ	$\mathcal{O}((n - \rho)^2) +$ contribution for \mathcal{A}_ρ	C.19, C.21
root is prime recursive, \mathcal{C} is satisfied, and T has size 2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	C.25
root is prime recursive (if not preceding case)	$\mathcal{O}(n - \rho) +$ contribution for \mathcal{A}_ρ	$\mathcal{O}(n - \rho) +$ contribution for \mathcal{A}_ρ	C.25, §C.5.2

Table 2: Complexities of the automata constructions, in all possible cases.

In the optimized version (resp. in the classical version) the complexity is at most in $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(n^4)$) in the non-recursive cases and at most in $\mathcal{O}((n - |\rho|)^2)$ plus the additional complexity of the construction of \mathcal{A}_ρ in the recursive cases. Notice that no extra time is needed to mark the states of the automaton, as they are marked when they are built. Consequently in the optimized version (resp. in the classical version) the automaton \mathcal{A}_π can be built in time and space $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(n^4)$), n being the size of $|\pi|$.

Indeed let K be the number of levels of recursion needed in the construction of \mathcal{A}_π . Then we can set $\rho_1 = \pi$ and define recursively permutations ρ_i for $2 \leq i \leq K$, ρ_i being the permutation ρ that appears recursively when building $\mathcal{A}_{\rho_{i-1}}$. From Table 2, we deduce that, in the optimized version, the time and

space complexity for building \mathcal{A}_π is:

$$\mathcal{O}((|\rho_1| - |\rho_2|)^2) + |\mathcal{A}_{\rho_2}| = \dots = \mathcal{O}\left(\sum_{i=1}^{K-1} (|\rho_i| - |\rho_{i+1}|)^2 + |\rho_K|^2\right).$$

Since every ρ_i is a pattern of π , we have $|\rho_i| - |\rho_{i+1}| \leq n$ and $|\rho_K| \leq n$. Hence, the time and space complexity for building \mathcal{A}_π is:

$$\mathcal{O}\left(n \cdot \left(\sum_{i=1}^{K-1} (|\rho_i| - |\rho_{i+1}|) + |\rho_K|\right)\right) = \mathcal{O}(n \cdot |\rho_1|) = \mathcal{O}(n^2).$$

In the same way we get the complexity $\mathcal{O}(n^4)$ for the classical version. \square

References

- [1] A. V. Aho, M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Comm. ACM*, 18 (1975) 333–340.
- [2] M. H. Albert, R. E. L. Aldred, M. D. Atkinson, D. A. Holton. Algorithms for pattern involvement in permutations, in: *ISAAC '01, LNCS 2223*, Springer, 2001, pp. 355–366.
- [3] M. H. Albert, M. D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Math.* 300 (2005) 1–15.
- [4] F. Bassino, M. Bouvel, A. Pierrot, C. Pivoteau, D. Rossin. Combinatorial specification of permutation classes, in: *FPSAC 2012, DMTCS proceedings AR*, 2012, pp. 781–792.
- [5] F. Bassino, M. Bouvel, A. Pierrot, D. Rossin. Deciding the finiteness of the number of simple permutations contained in a wreath-closed class is polynomial. *Pure Math. Appl. (PU.M.A.)* 21 (2010) 119–135.
- [6] F. Bassino, M. Bouvel, D. Rossin. Enumeration of pin-permutations. *Electron. J. Combin.* 18 (2011) Paper P57.
- [7] A. Bergeron, C. Chauve, F. de Montgolfier, M. Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. *SIAM J. Discrete Math.* 22 (2008) 1022–1039.
- [8] M. Bouvel, C. Chauve, M. Mishna, D. Rossin. Average-case analysis of perfect sorting by reversals. *Discrete Math. Algorithms Appl.* 3 (2011) 369–392.
- [9] R. Brignall, S. Huczynska, V. Vatter. Decomposing simple permutations, with enumerative consequences. *Combinatorica* 28 (2008) 385–400.
- [10] R. Brignall, S. Huczynska, V. Vatter. Simple permutations and algebraic generating functions. *J. Combin. Theory Ser. A* 115 (2008) 423–441.

- [11] R. Brignall, N. Ruškuc, V. Vatter. Simple permutations: decidability and unavoidable substructures. *Theoret. Comput. Sci.* 391 (2008) 150–163.
- [12] B.-M. Bui Xuan, M. Habib, C. Paul. Revisiting T. Uno and M. Yagiura’s algorithm, in: *ISAAC ’05*, LNCS 3827, Springer, 2005, pp. 146–155.
- [13] Ph. Flajolet, R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, Cambridge, 2008.
- [14] T. Gallai. Transitiv orientierbare Graphen, *Acta Math. Hungar.* 18 (1967) 25–66.
- [15] S. Heber, J. Stoye. Finding all common intervals of k permutations, in: *CPM 2001*, LNCS 2089, Springer, 2001, pp. 207–218.
- [16] J. E. Hopcroft, J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, USA, 1st edition, 1979.
- [17] D. E. Knuth. *Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading MA, 3rd edition, 1973.
- [18] A. Pierrot, D. Rossin. Simple permutation poset. Preprint available at <http://arxiv.org/abs/1201.3119>, 2014.