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

Frédérique Bassino[a], Mathilde Bouvel[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, Université de Bordeaux and CNRS, 351, cours de la Libération, 33405 Talence cedex, France.*
[c] *LIAFA UMR 7089, Université Paris Diderot and CNRS, Case 7014, 75205 Paris cedex13, 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.

1. Introduction

Since the definition of the pattern relation among permutations by Knuth in the 70's [18], the study of permutation patterns and permutation classes in combinatorics 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 (see [9, 13, 17] and their references among many others). Another line of research on permutation classes has been emerging for almost a decade: it is interested in properties or results that are less precise but

[1] This work was completed with the support of the ANR project ANR BLAN-0204_07 MAGNUM

apply to *families* of permutation classes. Examples of such general results may regard enumeration of permutation classes that fall into general frameworks [3, 4**?** , 19], properties of the corresponding generating functions [3, 10, 11, 12], growth rates of permutation classes [**?** **?** **?** **?** **?**], order-theoretic properties of permutation classes [**?** 5**?** **?**]... 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 [11, 10, 12]. 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 [3**?** , 6**?** , 11, 12**?** , 19], 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 [12] and [7].

In [12], 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, [12] 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 [12] 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 2.5.

Our goal is to solve the decision problem of [12] with an actual algorithm, whose complexity should be kept as low as possible. For this purpose, we heavily rely on [7] where we perform a detailed study of the class of *pin-permutations*, which contains the proper pin-permutations of [12]. These results allow us to precisely characterize the pin words corresponding to any given pin-permutation, and to subsequently modify the automata construction of [12], leading to our algorithm deciding whether a permutation class given by a finite basis B contains a finite number of simple permutations. 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 [12]. More precisely we give a $\mathcal{O}(n \log n + s^{4k})$ algorithm to decide if a finitely based permutation class $Av(\pi_1, \pi_2, \ldots, \pi_k)$ contains a finite number of simple permutations where $n = \sum |\pi_i|$ and $s \leq \max |\pi_i|$. We also describe a variant of our algorithm, whose complexity is $\mathcal{O}(n \log n + s^{2k})$. Notice that we described in [6] 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 $n = \sum |\pi_i|$.

The article is organized as follows. Section 2 is a reminder of previous definitions and results: permutation patterns, pin-permutations and their pin words, brief description of the characterization and the procedure of [12], decomposition trees, special families of pin-permutations playing an important role in our work. Section 3 recalls and refines the interpretation of the pattern containment relation between pin-permutations on their pin words. The main result of Section 3 is Theorem 3.13, which is the basis of the automata construction of Section 5. Section 4 focuses on the language of pin words encoding a given pin-permutation. Theorems 4.8, 4.12, 4.19, 4.25 and 4.31 describe, for any pin-permutation π the language $P(\pi)$ of pin words of π. These languages are described recursively following the recursive characterization of the decomposition trees of pin-permutations obtained in [7], and refining the ideas used in its proof. Then, based on these results and following the same recursive approach, Section 5 describes a recursive algorithm that builds, for any pin-permutation π, an automaton accepting the language $\mathcal{L}_\pi$ of pin words that encode proper pin-permutations containing π as a pattern (or rather, for technical reasons that will be explained later, a slight modification of this language). By Theorem 3.13, checking whether $\mathcal{C} = Av(B)$ contains finitely many proper pin-permutations is equivalent to checking whether the complement set of $\cup_{\pi \in B} \mathcal{L}_\pi$ is finite. Finally, Section 6 explains how to decide this question using the automata of Section 5, and combines this procedure with existing algorithms from [2, 6, 8, 12]: this provides an algorithm of reasonable complexity to decide, given a finite basis B, whether the class $\mathcal{C} = Av(B)$ contains a finite number of simple permutations. To conclude, we put this result in the context of previous and future research in Section 7.

2. Preliminaries

We recall in this section a few definitions and results about permutations, pin representations and pin words. More details can be found in [7, 11, 12].

2.1. Permutation classes and simple permutations

A permutation $\sigma \in S_n$ is a bijective function from $\{1, \ldots, n\}$ onto $\{1, \ldots, n\}$. We represent a permutation either by a word $\sigma = \sigma_1 \sigma_2 \ldots \sigma_n$ where $\sigma_i = \sigma(i)$ for every $i \in \{1, \ldots, n\}$, or by its *diagram* consisting in the set of points at

coordinates (i, σ_i) drawn in the plane. Figure 1 (p.5) shows for example the diagram of $\sigma = 4\,7\,2\,6\,3\,1\,5$. A permutation $\pi = \pi_1\pi_2\ldots\pi_k$ is a *pattern* of a permutation $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ and we write $\pi \leq \sigma$ if and only if there exist $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ such that π is isomorphic to $\sigma_{i_1}\ldots\sigma_{i_k}$. We also say that σ *involves* or *contains* π. If π is not a pattern of σ we say that σ *avoids* π.

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

A *block* (or *interval*) of a permutation σ of size n is a subset $\{i, \ldots, (i+\ell-1)\}$ of consecutive integers of $\{1, \ldots, n\}$ whose images by σ also form an interval of $\{1, \ldots, n\}$. 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 here.

Our goal is to check whether a permutation class contains a finite number of simple permutations, ensuring in this way that its generating function is algebraic [3]. In this article, permutation classes are given by their bases. We are further interested in classes having finite bases, since otherwise, from [3], they contain infinitely many simple permutations. As we shall see in the following, a class of particular permutations, called the pin-permutations, plays a central role in the decision procedure of this problem. For this reason, we record basic definitions and results related with these pin-permutations.

2.2. Pin-permutations and pin representations

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* of a set of points P is the smallest axis-parallel rectangle containing the set P. A *pin sequence* is a sequence $(p_1, \ldots, p_k)$ 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 $\{p_1, \ldots, p_{i-1}\}$ and satisfies one of the following conditions:

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

Lemma 2.17 of [7] is used several times in our proofs, and we state it here:

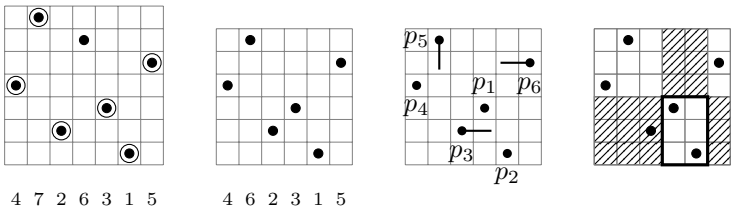


Figure 1: The permutation $\sigma = 4\,7\,2\,6\,3\,1\,5$, its pattern $\pi = 4\,6\,2\,3\,1\,5$, a pin representation p of π, and the bounding box of $\{p_1, p_2\}$ with its sides shaded.

Lemma 2.1. *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 2.2. *A pin representation of a simple pin-permutation is always proper as any independent pin p_i with $i \geq 3$ creates a block corresponding to $\{p_1, \dots, p_{i-1}\}$.*

2.3. Pin words

Pin representations can be encoded 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 2. The region encoded by 1 is called the first *quadrant* with respect to the box ■. The same goes for $2, 3, 4$. The letters U, D, L, R are called *directions*, while $1, 2, 3$ and 4 are *numerals*.

Example 2.3. *$14L2UR$ (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 $\pi = 4\,6\,2\,3\,1\,5$ shown in Figure 1.*

Example 2.3 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:

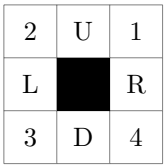


Figure 2: Encoding of pins by letters.

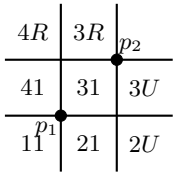


Figure 3: The two letters in each cell indicate the first two letters of the pin word encoding $(p_1, \ldots, p_n)$ when p_0 is taken in this cell.

Remark 2.4. *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 3). 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 2.5. *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.*

A *strict* (resp. *quasi-strict*) pin word is a pin word that begins with a numeral (resp. two numerals) followed only by directions.

Remark 2.6 (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.*

2.4. The first decision procedure of Brignall et al.

In [12], Brignall *et al.* study conditions for a class to contain a finite number of simple permutations. Introducing three new kinds of permutations they show that this problem is equivalent to ensuring that the class contains a finite number of permutations of each of these three simpler kinds.

Theorem 2.7. *[10, 12] A permutation class $Av(B)$ 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.*

In Theorem 2.7 above, the proper pin sequences of [12] have been replaced by proper pin-permutations, which is equivalent. Indeed, containing a finite number of proper pin-permutations is equivalent to containing a finite number of proper pin sequences, since the encoding of proper pin-permutations by proper pin sequences provides a finite-to-one correspondence. More precisely, each proper pin sequence corresponds to a unique proper pin-permutation. Conversely every proper pin-permutation of size n is associated with at least one and very loosely at most 8^n pin sequences, since pin sequences are encoded by pin words on an 8-letter alphabet.

The definition of the wedge simple permutations and the parallel alternations are not crucial to our work, hence we refer the reader to [12] for more details. What is however important for our purpose is to be able to test whether a class given by a finite basis contains a finite number of parallel alternations and wedge simple permutations.

Alternations and wedge simple permutations, that can be of type 1 or 2, are well characterized in [12]. This characterization leads to the following lemmas:

Lemma 2.8. *[12] 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 2.9. *[12] 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 2.10. *[12] 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 2.11. *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 2.8 to 2.10, 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 length at most 4 can be done in $\mathcal{O}(|\pi| \log |\pi|)$. As we have to 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. □

In [12] 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 language theoretic arguments. We review its structure in the next subsection, so that we may analyze it and compare it with the work we present in this article.

2.5. Towards an actual algorithm

In [12], the authors define an order relation $\preccurlyeq$ on pin words (whose definition we recall p.13). Denoting by $\mathcal{SP}$ the set of all strict pin words, and by $P(B)$ the set of pin words encoding a permutation of B, they prove that $\mathcal{C} = Av(B)$ contains a finite number of proper pin-permutations if and only if the set $\mathcal{L} = \mathcal{SP} \setminus \bigcup_{u \in P(B)} \{\text{strict pin word } w \mid u \preccurlyeq w\}$ is finite. Then given a pin word u, they explain how to build an automaton $\mathcal{A}^{(u)}$ recognizing a language $\mathcal{L}^{(u)}$ such that $\mathcal{SP} \cap \mathcal{L}^{(u)} = \{\text{strict pin word } w \mid u \preccurlyeq w\}$. Finally, they notice that $\mathcal{SP}$ is a recognizable language, and conclude – referring to 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 [12] 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 [12], 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 [12], 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 [12], 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 [12] 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 size $|\pi|$, 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, [12] 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, [12] 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, [12] 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 6.3, as our algorithm ends with a similar step).

The goal of the present work is to give an actual algorithm of complexity as small as possible to determine whether a permutation class $\mathcal{C}$ given by a finite basis B contains a finite number of simple permutations. To do so, we complete and modify the procedure of [12] and obtain an algorithm whose total complexity (with the same notations as above) is $\mathcal{O}(n\log n + s^{2k})$, which is reasonable to be used in practice.

Like in [12], the main difficulty is to decide whether $\mathcal{C}$ contains a finite number of proper pin-permutations, and this decision problem is also solved using automata theory. We first compute a description of the set $P(B)$ (instead of considering $P(B)$ one element after another) and then build on automaton depending on $P(B)$ in which we look for a cycle. Thanks to a detailed study of pin-permutations in [7], which we further refine in Section 4, the description of $P(B)$ is computed in time $\mathcal{O}(n)$ and then the automaton is built in time $\mathcal{O}(s^{2k})$. This is to be compared with the respective complexities $\mathcal{O}(n\cdot 8^{s'})$ and $\mathcal{O}(2^{k\cdot s\cdot 2^s})$ in the four-step procedure deduced from the work of [12]. Writing that $\mathcal{O}(s^{2k}) = \mathcal{O}(2^{k\cdot 2\log s})$ enable to measure the complexity improvement w.r.t. $\mathcal{O}(2^{k\cdot s\cdot 2^s})$: the complexity gain is doubly exponential w.r.t. s.

2.6. Decomposition trees

Let σ be a permutation of S_k and $\pi^{(1)},\dots,\pi^{(k)}$ be k permutations of S_{p_1}, …, S_{p_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 $\pi^{(i)}$. More formally, this can be expressed by

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

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

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

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

Permutations can be decomposed using substitutions, as described in Theorem 2.13 below. For this purpose, we now introduce some definitions and notations. For any $n \geq 2$, let I_n be the permutation $1\,2 \ldots n$ and D_n be $n\,(n-1)\ldots 1$. Denote by $\oplus$ and $\ominus$ respectively I_n and D_n. Notice that in inflations of the form $\oplus[\pi_1, \pi_2, \ldots, \pi_n] = I_n[\pi_1, \pi_2, \ldots, \pi_n]$ or $\ominus[\pi_1, \pi_2, \ldots, \pi_n] = D_n[\pi_1, \pi_2, \ldots, \pi_n]$, the integer n is determined without ambiguity by the number of permutations π_i of the inflation.

Definition 2.12. *A permutation σ is* $\oplus$-decomposable *(resp.* $\ominus$-decomposable*) if it can be written as $\oplus[\pi_1, \pi_2, \ldots, \pi_k]$ (resp. $\ominus[\pi_1, \pi_2, \ldots, \pi_k]$), for some $k \geq 2$. Otherwise, it is* $\oplus$-indecomposable *(resp.* $\ominus$-indecomposable*).*

Theorem 2.13. ([3], first appeared implicitly in [15]) *For any $n \geq 2$, every permutation $\sigma \in S_n$ can be uniquely decomposed as either:*

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

It is important for stating Theorem 2.13 that 12 and 21 are not considered as simple permutations. An equivalent version of this theorem, which includes 12 and 21 among simple permutations, is given in [3]. Another important remark is that:

Remark 2.14. *Any block of $\sigma = \alpha[\pi_1, \ldots, \pi_k]$ (with α a simple permutation) is either σ itself, or is included in one of the π_i.*

Theorem 2.13 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 $\alpha[\pi_1, \ldots, \pi_k]$ is represented by a node labeled α with k ordered children representing the π_i.

Definition 2.15. *The substitution decomposition tree T of the permutation σ is the unique labeled ordered tree encoding the substitution decomposition of σ, where each internal node is either labeled by $\oplus, \ominus$ – those nodes are called* linear *– or by a simple permutation α –* prime *nodes. Each node labeled by α has arity $|\alpha|$.*

Example 2.16. *Let $\sigma = 10\,13\,12\,11\,14\,1\,18\,19\,20\,21\,17\,16\,15\,4\,8\,3\,2\,9\,5\,6\,7$. Its substitution decomposition (see also Figure 4) can be written as:*
$3\,1\,4\,2[\oplus[1, \ominus[1, 1, 1], 1], 1, \ominus[\oplus[1, 1, 1, 1], 1, 1, 1], 2\,4\,1\,5\,3[1, 1, \ominus[1, 1], 1, \oplus[1, 1, 1]]]$.

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 2.13.

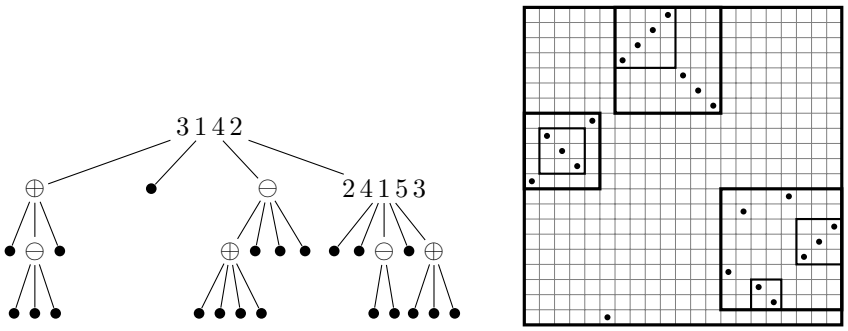


Figure 4: The diagram and the substitution decomposition tree T of permutation $\sigma = 10\,13\,12\,11\,14\,1\,18\,19\,20\,21\,17\,16\,15\,4\,8\,3\,2\,9\,5\,6\,7$. The internal nodes of T correspond to the blocks of σ marked by rectangles.

Remark 2.17. *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.

2.7. Oscillations and quasi-oscillations

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

Following [12], let us consider the infinite oscillating sequence defined (on $\mathbb{N}\backslash\{0,2\}$ for regularity of the diagram) by $\omega = 4\,1\,6\,3\,8\,5\,\ldots(2k{+}2)\,(2k{-}1)\,\ldots$. Figure 5 shows the diagram of a prefix of ω.

Definition 2.18 (oscillation)**.** *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 reverse*[2] *of an increasing oscillation.*

As noticed in [7] there are two increasing (resp. decreasing) oscillations of size n for any $n \geq 3$. 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.

Definition 2.19 (quasi-oscillations [7])**.** *An* increasing quasi-oscillation *of size $n \geq 6$ is obtained from an increasing oscillation ξ of size $n-1$ by the addition of either a minimal element at the beginning of ξ or a maximal element at the end of ξ, followed by a* flip *of an element of ξ according to the rules of Table 1*[3]*.*

[2]The reverse of $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ is $\overleftarrow{\sigma} = \sigma_n\ldots\sigma_2\sigma_1$

[3] 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 flipping the left-most point of ξ to the right-most (in β), and the main substitution point is the largest point of ξ (see Figure 5).

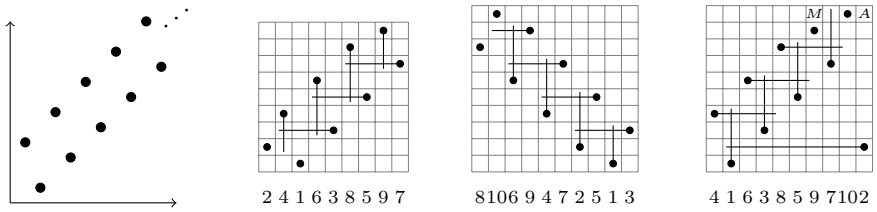


Figure 5: The infinite oscillating sequence ω, an increasing oscillation ξ of size 9, a decreasing oscillation of size 10, and the increasing quasi-oscillation of size 10 obtained from ξ by addition of a maximal element, with a pin representation for each.

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: 2 4 1 3*,* 3 1 4 2*,* 2 5 3 1 4 *and* 4 1 3 5 2*. Each of them has two possible choices for its pair of auxiliary and main substitution points. See Figure 6 for more details. Finally, a* decreasing quasi-oscillation *is the reverse of an increasing quasi-oscillation.*

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

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

As noticed in [7] 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). 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. These remarks will be useful in the following.

Remark 2.20. *Oscillations of size at least* 4 *and quasi-oscillations are simple pin-permutations.*

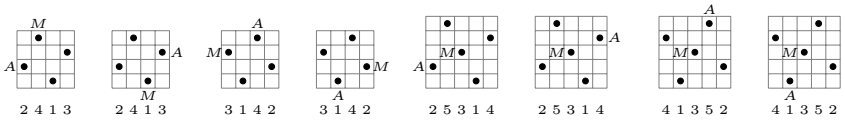


Figure 6: The diagrams of the increasing quasi-oscillations of size 4 and 5.

We refer the reader to [7] for further properties of oscillations and quasi-oscillations.

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

First, following [12], we interpret the pattern containment relation on pin-permutations by a piecewise factor relation between their pin words. This interpretation then leads to the characterization of the relation $\pi \leq \sigma$, for σ a pin-permutation, by a set inclusion of the form $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$ for languages we introduce in the following. Subsequently, taking $\pi \in B$, we show how these languages $\mathcal{L}_\pi$ can be used to characterize permutation classes $Av(B)$ that contain finitely many proper pin-permutations.

3.1. Pattern containment and piecewise factor relation

Recall the definition of the partial order $\preccurlyeq$ on pin words introduced in [12].

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

- *if $w^{(i)}$ begins with a numeral then $w^{(i)} = u^{(i)}$, and*
- *if $w^{(i)}$ begins with a direction, then $v^{(i)}$ is nonempty, the first letter of $w^{(i)}$ corresponds to a point lying in the quadrant – 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 3.2. *The strong numeral-led factor decomposition of $u = 14L2UR$ is $u = 1 \cdot 4L \cdot 2UR$. Moreover, $u \preccurlyeq w = 2RU4LULURD4L$, because w may be decomposed as $w = 2R\mathbf{U} \cdot \mathbf{4L} \cdot U\mathbf{LUR} \cdot D4L$, where the factors $w^{(i)}$ satisfying the conditions of Definition 3.1 being emphasized by bold letters.*

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

Lemma 3.3. *[12] If the pin word w encodes the permutation σ and $\pi \leq \sigma$ then there is a pin word u encoding π with $u \preccurlyeq w$. Conversely if $u \preccurlyeq w$ then the permutation corresponding to u is contained in the one corresponding to w.*

In what follows, σ is a proper pin-permutation (recall that our goal is to test whether there are finitely many proper pin-permutations in a class). So we can choose a strict pin word w_σ that encodes σ (see Remark 2.6 p.6). As a consequence of Lemma 3.3, 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 $u \preccurlyeq w_\sigma$.

The relation $u \preccurlyeq w$ on pin words is nearly a piecewise factor relation, the factors being determined by the strong numeral-led factors of u. We use an encoding of pin words that we introduced in [6] and recall hereafter: it maps the relation $\preccurlyeq$ on an actual piecewise factor relation. As explained above, for our purpose it will be enough to consider in Lemma 3.12 the case where w is a strict pin word. Denote by $\mathcal{SP}$ the language of strict pin words. Let $\mathcal{SP}_{\geq 2} = \mathcal{SP} \setminus \{1, 2, 3, 4\}$ be the set of strict pin words of length at least 2. 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 in replacing the only numeral in a strict pin word by two directions. Intuitively, given a numeral q and a box ■, 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 ■.

Definition 3.4. *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 $u = u_1 u_2 \ldots \in \mathcal{SP}_{\geq 2}$, $u_i = \phi(u)_{i+1}$ 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 3.5 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 3.5. *Let w be a pin word and p be the pin representation corresponding to w. For any $i \geq 2$, set*

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

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

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 3.4.

Proof. Similar to the proof of Lemma 3.4 of [6], adapted to the case where w is any pin word, *i.e.*, is not necessarily strict. □

Lemma 3.5 is used in the proofs of Lemma 3.7 and Theorem 3.8. Their statement also requires that we extend some definitions from $\mathcal{SP}$ to $\mathcal{M}$.

Remark 3.6. *Words of $\mathcal{M}$ may also be seen as encodings of pin sequences (as in Subsection 2.3), taking the origin p_0 to be a box instead of a point. Moreover, the relation $u \preccurlyeq w$ can be extended to $w \in \mathcal{M}$, and the map ϕ can be defined on words of $\mathcal{M}$ as the identity map.*

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

Lemma 3.7. *Let u be a strict pin word and w be a word of $\mathcal{SP} \cup \mathcal{M}$. If $|u| \geq 2$ then $u \preccurlyeq w$ if and only if $\phi(u)$ is a factor of $\phi(w)$. If $|u| = 1$ then $u \preccurlyeq w$ if and only $\phi(w)$ 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 to Lemma 3.5 of [6]. Other cases are proved in a similar way. □

In the statement of Lemma 3.7, 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 3.7 can be extended formalizing the idea of piecewise factors mentioned at the beginning of this section.

Theorem 3.8. *Let u and w be two pin words and $u = u^{(1)} \ldots u^{(j)}$ be the strong numeral-led factors decomposition of u. Then $u \preccurlyeq w$ if and only if w can be chopped into a sequence of factors $w = v^{(1)}w^{(1)} \ldots v^{(j)}w^{(j)}v^{(j+1)}$ such that for all $i \in \{1, \ldots, j\}$, $w^{(i)} \in \mathcal{SP} \cup \mathcal{M}$ and $\phi(w^{(i)})$ has a factor in $\phi(u^{(i)})$.*

Proof. We prove that $u \preccurlyeq w$ 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 $u^{(i)} \preccurlyeq w^{(i)}$. Then the result follows using Lemma 3.7.

If $u \preccurlyeq w$, 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 3.1. 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 $u^{(i)} \preccurlyeq w^{(i)}$. Indeed, if $\bar{w}^{(i)}$ begins with a direction, by Lemma 3.5 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 $i \in \{1, \dots, j\}$, $u^{(i)} \preccurlyeq w^{(i)}$ then from Definition 3.1 we can decompose $w^{(i)}$ as $y^{(i)}\bar{w}^{(i)}z^{(i)}$ and thanks to Lemma 3.5 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 3.1. □

3.2. Pattern containment and set inclusion

Definition 3.9. *Let u be a pin word and $u = u^{(1)} \dots u^{(j)}$ be its strong numeralled factor decomposition. We set*

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

where $A = \{U, D, L, R\}$ (or $A = \{U, D, L, R, \sharp\}$ in Theorem 3.10(i)).
Let π be a permutation, and $P(\pi)$ be the set of pin words that encode π. We set

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

Note that $\mathcal{L}_\pi$ is nonempty if and only if $P(\pi)$ is nonempty or equivalently π is a pin-permutation. When π is not a pin-permutation, the results of Theorem 3.10 and Lemma 3.12 follow easily from the following statement (see for instance Lemma 3.3 of [7]): if $\pi \leq \sigma$ and σ is a pin-permutation, then π is a pin-permutation.

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 $m \in A^\star$, 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.

The languages $\mathcal{L}_\pi$ have been introduced in Definition 3.9 because they describe the proper pin-permutations that contain π. Indeed $\mathcal{L}_\pi \cap \mathcal{M}$ is in one-to-one correspondence with pin words encoding proper pin-permutations that contain π, via ϕ^{-1}. Moreover the languages $\mathcal{L}_\pi$ somehow translate the pattern involvement between pin-permutations into set inclusion:

Theorem 3.10. *Let π and σ be permutations, σ being a pin-permutation. Then*

(*i*) *for $A = \{U, D, L, R, \sharp\}$, $\pi \leq \sigma$ if and only if $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$;*

(*ii*) *for $A = \{U, D, L, R\}$, if $\pi \leq \sigma$ then $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$.*

Remark 3.11. *In the sequel, we always take $A = \{U, D, L, R\}$, and use only the second statement of Theorem 3.10. However, we find interesting to have an equivalence between pattern containment between pin-permutations and set inclusion in the first statement of this theorem with $A = \{U, D, L, R, \sharp\}$. We do not know if the equivalence still holds when $A = \{U, D, L, R\}$.*

Proof. Suppose that $\pi \leq \sigma$. Let $m \in \mathcal{L}_\sigma$, we want to show that $m \in \mathcal{L}_\pi$. By definition of $\mathcal{L}_\sigma$, 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 3.3 there is $u \in P(\pi)$ such that $u \preccurlyeq w$. Let $u = u^{(1)} \dots u^{(j)}$ be the strong numeral-led factor decomposition of u. From Theorem 3.8, $w = \bar{v}^{(1)}\bar{w}^{(1)} \dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$ 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 numeral-led 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 $\phi(\bar{w}^{(1)})$, $\phi(\bar{w}^{(2)})$, …, $\phi(\bar{w}^{(j)})$ 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 $m = v^{(1)}\phi(w^{(1)})v^{(2)}\phi(w^{(2)}) \dots v^{(i)}\phi(w^{(i)})v^{(i+1)} \in A^\star\phi(\bar{w}^{(1)})A^\star\phi(\bar{w}^{(2)}) \dots A^\star\phi(\bar{w}^{(j)})A^\star$. But for all $k \in \{1, \dots, j\}$, $\phi(\bar{w}^{(k)})$ has a factor in $\phi(u^{(k)})$, thus $m \in \mathcal{L}(u) = A^\star\phi(u^{(1)})A^\star\ \phi(u^{(2)}) \dots A^\star\phi(u^{(j)})A^\star$ and so $m \in \mathcal{L}_\pi$.

Conversely, in the case $A = \{U, D, L, R, \sharp\}$, if $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$, let us show that $\pi \leq \sigma$. From Lemma 3.3 it is sufficient to show that there is some $u \in P(\pi)$ and some $w \in P(\sigma)$ such that $u \preccurlyeq w$. As σ is a pin-permutation, $P(\sigma)$ is not empty. Let w be a word of $P(\sigma)$ and $w^{(1)}w^{(2)} \dots w^{(i)}$ its strong numeral-led factor decomposition. Let $m = \phi(w^{(1)})\,\sharp\,\phi(w^{(2)}) \dots \,\sharp\,\phi(w^{(i)})$. Obviously, $m \in \mathcal{L}(w)$ so that $m \in \mathcal{L}_\sigma$ and $m \in \mathcal{L}_\pi$. By definition of $\mathcal{L}_\pi$ there is some $u \in P(\pi)$ such that $m = v^{(1)}\phi(u^{(1)})v^{(2)}\phi(u^{(2)}) \dots \phi(u^{(j)})v^{(j+1)}$ where $u^{(1)}u^{(2)} \dots u^{(j)}$ is the strong numeral-led factor decomposition of u. But $m = \phi(w^{(1)})\,\sharp\,\phi(w^{(2)}) \dots \,\sharp\,\phi(w^{(i)})$ and there is no letter $\sharp$ in the $\phi(u^{(k)})$. So the factors $\phi(u^{(1)})$, $\phi(u^{(2)})$, $\dots \phi(u^{(j)})$ appear in this order in $\phi(w^{(1)})\phi(w^{(2)}) \dots \phi(w^{(i)})$, being non-overlapping and each inside one $\phi(w^{(k)})$. Therefore w can be chopped into a sequence of factors $w = \bar{v}^{(1)}\bar{w}^{(1)} \dots \bar{v}^{(j)}\bar{w}^{(j)}\bar{v}^{(j+1)}$ such that 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 $\phi(u^{(k)})$. Thus from Theorem 3.8 $u \preccurlyeq w$, concluding the proof. □

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

We conclude Section 3 by explaining how the above definitions and results are related to our original problem: testing whether a permutation class contains finitely many proper pin-permutations.

Lemma 3.12. *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 3.10(*ii*) $\mathcal{L}_\sigma \subseteq \mathcal{L}_\pi$. 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 $t^{(i)} \in A^\star \phi(u^{(i)}) \cap \mathcal{M}$ 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 3.8, $u \preccurlyeq w$. Finally from Lemma 3.3, we conclude that $\pi \leq \sigma$. □

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 3.12, $\mathcal{L}_\pi \cap \mathcal{M}$ is the image by ϕ of the language of strict pin words encoding proper pin-permutations σ that contain π as a pattern: $\mathcal{L}_\pi \cap \mathcal{M} = \{\phi(w) \mid \exists \sigma$ such that $\pi \leq \sigma$ and $w \in \mathcal{SP} \cap P(\sigma)\}$. With the same idea we have the following theorem:

Theorem 3.13. *A permutation class $Av(B)$ contains a finite number of proper pin-permutations if and only if the set $\mathcal{M} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$ is finite.*

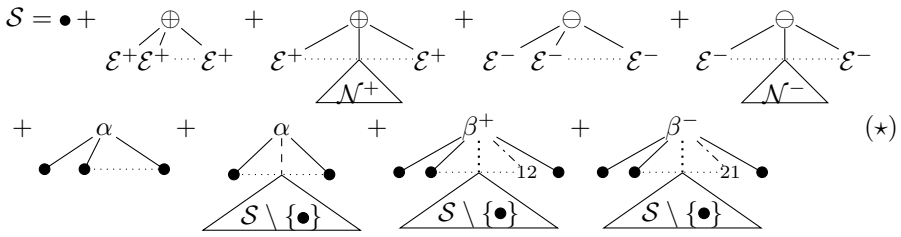
Proof. Let S_B be the set of strict pin words encoding permutations of size at least 2 of $Av(B)$. Then ϕ is a bijection from S_B to $\mathcal{M}_{\geq 3} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$. Indeed for any strict pin word w of size at least 2, let σ be the permutation encoded by w. Then σ is a proper pin-permutation and Lemma 3.12 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 at most 8^n strict pin words, as any pin word encoding σ is a word of length n over an 8-letter alphabet. □

From Theorem 3.13, our goal is now to find an algorithm checking if $\mathcal{M} \setminus \cup_{\pi \in B} \mathcal{L}_\pi$ is finite, given a finite basis B. To do this, in Section 5 we construct automata recognizing $\mathcal{L}_\pi$ for any pin-permutation π. As the language $\mathcal{L}_\pi$ is defined from the set $P(\pi)$ of pin words of π, we first need to explicitly describe $P(\pi)$ for any pin-permutation π. This is done in Section 4.

4. Pin words of pin-permutations

In this section, our goal is to describe the set $P(\pi)$ of pin words that encode a pin-permutation π. In [7], a recursive characterization of the decomposition trees of pin-permutations is provided, and we follow it to recursively describe $P(\pi)$. The characterization of [7] is as follows. The set $\mathcal{S}$ of substitution de-

composition trees of pin-permutations is recursively characterized by:



where $\mathcal{E}^+$ (resp. $\mathcal{E}^-$) is the set of decomposition trees of increasing (resp. decreasing) oscillations (see Definition 2.18 p.11), $\mathcal{N}^+$ (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. Edges written _ _ _ (resp. _._._, ...) correspond to an active point – see p.24 – of α (resp. to a pair formed by an auxiliary point and a main substitution point of β). In this definition the only terms that are recursive are those containing a subtree labeled by $\mathcal{N}^+, \mathcal{N}^-$ or $\mathcal{S} \setminus \{\bullet\}$.

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 4.1. *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.*

4.1. Reading of children of a linear node

Definition 4.2. *Let π be a pin-permutation and $p = (p_1, \ldots, p_n)$ be a pin representation of π. For any set D of points of π, if k is the number of maximal factors $p_i, p_{i+1}, \ldots, p_{i+j}$ 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 before *(resp.* read entirely before*) C if the first pin in D (resp. 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 $T = \overset{\oplus}{(T_1\ T_2\ \cdots\ T_r)}$ and let $p = (p_1, \ldots, p_n)$ be one of its pin representations. In the sequel, we denote by i_0 the index of the child which contains p_1.

Lemma 4.3. *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 $\mathcal{B}_{p_1,\ldots,p_\ell}$ be the bounding box of $\{p_1, \ldots, p_\ell\}$. As $p_1 \in T_{i_0}$ and $p_\ell \in T_i$, $T_j \subseteq \mathcal{B}_{p_1,\ldots,p_\ell}$ (see Figure 7), hence it is entirely read before p_ℓ in p. Indeed, for all $k \geq 2$, p_k lies outside the bounding box of $\{p_1, \ldots, p_{k-1}\}$. □

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

Proof. Suppose that pins $p_1, \ldots, p_k$ 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 4.3 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 $i = \min\{\ell \mid \exists h \in \{1, \ldots, k\}, p_h \in T_\ell\}$. The same goes for m changing the minimum into a maximum. If the reading of T_i is not finished, since T_i is $\oplus$-indecomposable, there must exist a pin p_q in zone ▨ (see Figure 8). Such a pin is on the side of the bounding box $\mathcal{B}_{p_1,\ldots,p_k}$ of $\{p_1, \ldots, p_k\}$, and the same remark goes for T_m. But from Lemma 2.1 (p.5) there is at most one pin lying on the sides of a bounding box, and this contradiction concludes the proof. □

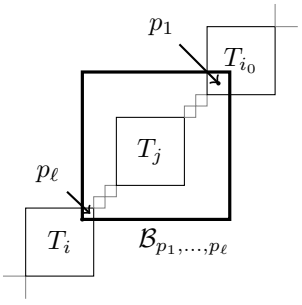


Figure 7: Proof of Lemma 4.3.

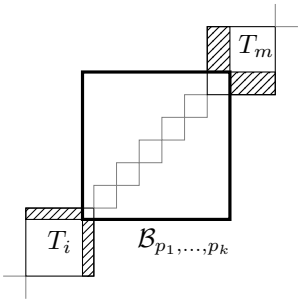


Figure 8: Proof of Lemmas 4.4 and 4.5.

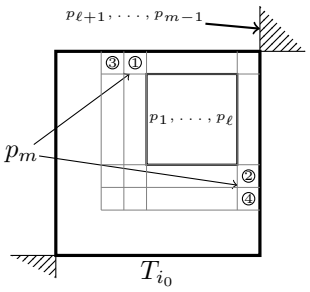


Figure 9: T_{i_0} is read in two pieces (Lemma 4.6).

Lemma 4.5. *Every child T_i is read in one time 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 time by p. Consider the pin p_{k+1} which is the first pin outside T_i after p has started reading T_i. As p_1 is in T_{i_0}, p_1 is outside T_i and the bounding box

of $\{p_1, p_2, \ldots, p_{k-1}, p_k\}$ allows to define a zone ▨ in T_i as shown in Figure 8. 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 $\{p_1, p_2, \ldots, p_k\}$ so it is p_{k+1} by Lemma 2.1 (p.5). Thus $p_{k+1} \in T_i$ which provides the desired contradiction. □

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 4.6. *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 10 where ξ^+ is an increasing oscillation of size at least 4.*

A given permutation π may match several shapes of Figure 10. However if a child is read in more than one time, 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 10 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.

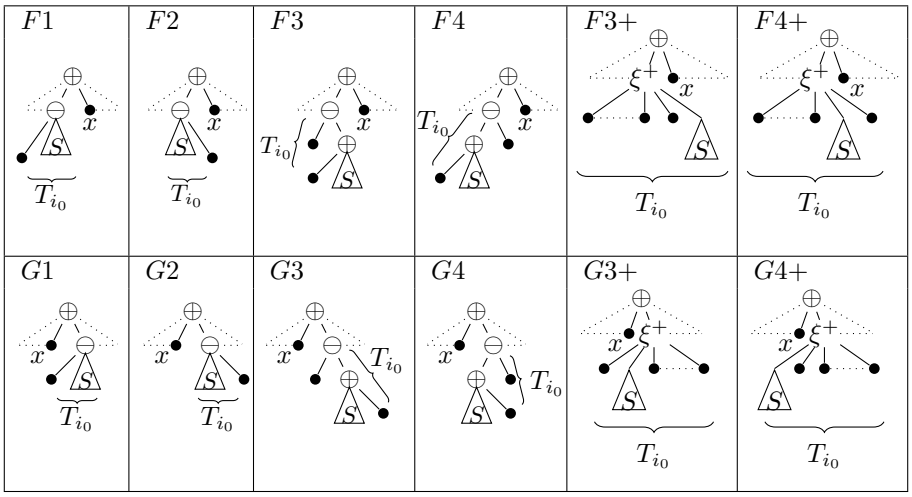
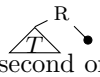


Figure 10: Decomposition tree of π when T_{i_0} may be read in several pieces.

In Figure 10 and in the sequel, we draw the attention of the reader to the difference between trees of the shape  and  : 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 $p = (p_1, p_2, \ldots, p_n)$ be a pin representation of π that reads one child in several pieces. Lemma 4.5 ensures that there is only one such child, which is necessarily T_{i_0}. Denote $p_1, \ldots, p_\ell$ the first part of the reading of T_{i_0}. Then $p_{\ell+1}, \ldots, p_{m-1}$ 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 $i \in \{\ell+1, \ell+2, \ldots, m-1\}$ lies in one of the zones ▨ as shown in Figure 9. But if both zones contain at least one pin p_i with $i \in \{\ell+1, \ell+2, \ldots, m-1\}$, the bounding box of $\{p_1, \ldots, p_{m-1}\}$ contains T_{i_0} and thus p_m cannot respect the externality condition. Hence all pins p_i with $i \in \{\ell+1, \ell+2, \ldots, m-1\}$ are in the same zone.

Assume w.l.o.g. that $\{p_{\ell+1}, \ldots, p_{m-1}\}$ are in the upper right zone of Figure 9 (otherwise, in the proof that follows, cases $F1, \ldots, F4+$ of Figure 11 are replaced by cases $G1, \ldots, G4+$). If p_m respects the independence condition, it must lie in the lower left corner of the bounding box of $\{p_1, \ldots, p_\ell\}$ 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 $m = \ell + 2$.

As only one point can lie on the sides of a bounding box, there are only four possible positions for p_m as depicted in Figure 9. If there is no pin separating p_m from $\{p_1, \ldots, p_{m-1}\}$ then p_m is either in position ① (case $F1$ on Figure 11) or ② (case $F2$); moreover pins $\{p_1, p_2, \ldots, p_\ell, p_m\}$ 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 $\{p_1, \ldots, p_\ell\}$, 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 $\{p_1, \ldots, p_m\}$ 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, $p_1, \ldots p_\ell$ for the first part and $p_m, \ldots, p_{m+k}$ with $m = \ell + 2$ for the second part. And from Lemma 4.5 the pin $p_{\ell+1}$ is by itself a child T_i of the root. It is then straightforward to check from Figure 11 that π has a decomposition tree of one shape given in Figure 10 with $S = \{p_1, \ldots, p_\ell\}$ and $x = p_{\ell+1}$. □

Note that in the proof of Lemma 4.6, 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 4.7. *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.19).

4.2. Non-recursive cases

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

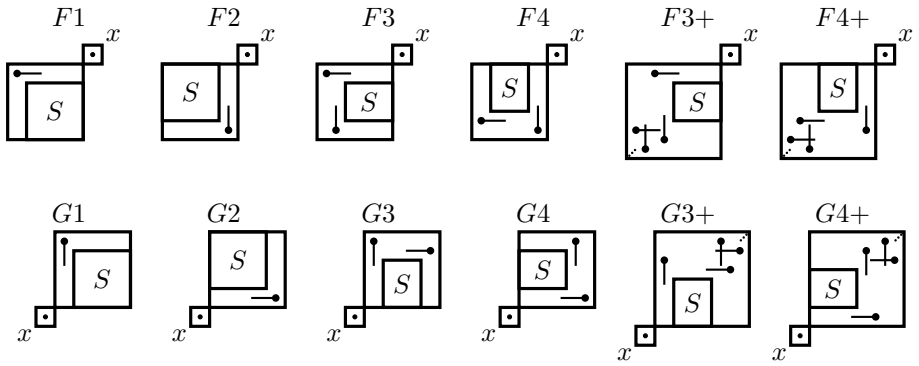
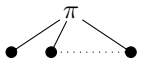


Figure 11: Diagram of T_{i_0} and x if T_{i_0} is read in two pieces, the first part being S.

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 4.8. *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 2.2 p.5) and $|\pi| \geq 3$, so p_3 is a separating pin and p is associated to 6 pin words by Remark 2.4 (p.6).

Moreover by Lemmas 4.3 and 4.6 of [7] 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 2.6 (p.6) ensures that they are all strict or quasi-strict. □

Theorem 4.8 of course does not describe the set $P(\pi)$ of pin words encoding a simple pin-permutation π explicitly, but Algorithm 2 of [6] explains how to compute $P(\pi)$ in this case. We will get back to this computation in Section 6.

Permutations whose decomposition trees have a linear root. W.l.o.g., since we focus on the non-recursive case, $\pi = \xi_1 \xi_2 \cdots \xi_r$ where ξ_i are increasing oscillations (see Remark 4.1 p.19). Lemma 4.9 is a direct consequence of Lemma 4.6.

Lemma 4.9. *Let $p = (p_1, p_2, \ldots, p_n)$ 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 time by p.

Lemma 4.9 together with Lemma 4.3 leads to the following.

Consequence 4.10. *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 time 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 4.10 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). Therefore we introduce the following functions $P^{(h)}$: 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 4.10 leads us naturally to introduce the shuffle product of sequences. From the above discussion, Theorem 4.12 then follows, providing the desired characterization.

Definition 4.11. *Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2, \dots, \mathsf{A}_q)$ and $\mathsf{B} = (\mathsf{B}_1, \mathsf{B}_2, \dots, \mathsf{B}_s)$ be two sequences of sets of words. The* shuffle product *$\mathsf{A} \sqcup\!\sqcup \mathsf{B}$ of A and B is defined as*

$$\begin{aligned}\mathsf{A} \sqcup\!\sqcup \mathsf{B} \quad &= \big\{c = c_1 \cdot \ldots \cdot c_{q+s} \mid \exists\, I = \{i_1, \dots, i_q\}, J = \{j_1, \dots, j_s\},\, I \cap J = \emptyset \\ &\qquad \text{with } i_1 < \ldots < i_q, j_1 < \ldots < j_s,\, (c_{i_1}, \dots, c_{i_q}) \in \mathsf{A},\, (c_{j_1}, \dots, c_{j_s}) \in \mathsf{B}\big\}.\end{aligned}$$

Theorem 4.12. *The set $P(\pi)$ of pin words of a permutation $\pi = \oplus[\xi_1, \dots, \xi_j]$ where every ξ_i is an increasing oscillation is:*

$$P(\pi) = \bigcup_{1 \le i \le r-1} P(\oplus[\xi_i, \xi_{i+1}]) \cdot \Big((P^{(1)}(\xi_{i-1}), \dots, P^{(1)}(\xi_1)) \sqcup\!\sqcup (P^{(3)}(\xi_{i+2}), \dots, P^{(3)}(\xi_j)) \Big).$$

Lemmas 4.13 and 4.18 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 4.12 an explicit expression for $P(\pi)$. Note that similar results can be obtained for permutations with root $\ominus$ and decreasing oscillations using Remark 4.1 (p.19).

We first recall a few definitions and facts about increasing oscillations.

An *active knight* of a permutation is a pair of pins that is a possible start of a pin representation of the permutation and an *active point* is a point belonging to an active knight. Lemma 4.6 of [7] describes the active knights of simple pin-permutations. In particular, 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 12).

These active knights are either in horizontal (H) position  or in vertical (V) position  . Therefore there are four types of increasing oscillations: (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 12). Note that an even size oscillation has type (H, H) or (V, V) and an odd size one (H, V) or (V, H). An comprehensive study of the different cases illustrated in Figure 12 leads to the following statement.

Lemma 4.13. *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:

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

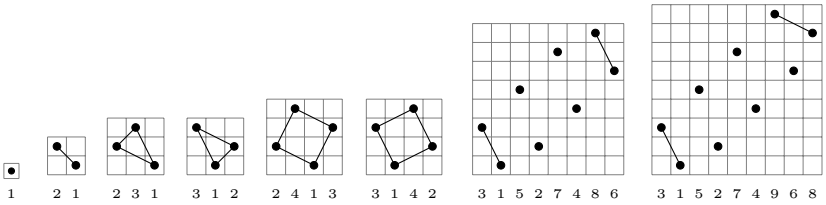


Figure 12: The increasing oscillations of size less than 5 and two increasing oscillations respectively of size 8 with type (V, V) and 9 with type (V, H). Active knights are marked by edges between their two active points.

Remark 4.14. *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 3.4 p.14), and for any increasing oscillation ξ, we have $\phi(P^{(3)}(\xi)) \subseteq \{U, R\}^\star$ and $\phi(P^{(1)}(\xi)) \subseteq \{L, D\}^\star$.*

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 4.18. 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 4.15. *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:

$P(1) = \{1, 2, 3, 4\}$ $\quad$ $P(21) = Q^- + S_H^- + S_V^-$

$P(231) = (Q^- + S_H^-) \cdot U + (Q^- + S_V^-) \cdot L + (Q^+ + S_H^+ + S_V^+) \cdot 4$

$P(312) = (Q^- + S_H^-) \cdot D + (Q^- + S_V^-) \cdot R + (Q^+ + S_H^+ + S_V^+) \cdot 2$

$P(2413) = (Q^- + S_H^-) \cdot (UR + DL) + (Q^+ + S_V^+) \cdot (RD + LU)$

$P(3142) = (Q^+ + S_H^+) \cdot (UL + DR) + (Q^- + S_V^-) \cdot (RU + LD)$

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 12. □

Definition 4.16. *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 4.17. *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

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

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

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

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 4.9 (p.24) 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 4.6 (p.21) the decomposition tree of $\oplus[\xi_i,\xi_j]$ has one of the shapes of Figure 10 (p.21), 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 10 and 11). 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 2.4 p.6) are those given in Lemma 4.17.

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 4.6 and Remark 4.7 (p.22) 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 4.17. □

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 4.18. *For any pair of increasing oscillations (ξ_i,ξ_j):*

- *If $|\xi_i| > 1$ and $|\xi_j| > 1$, $P(\oplus[\xi_i,\xi_j]) = \big(P(\xi_j)\cdot P^{(1)}(\xi_i)\big) \bigcup \big(P(\xi_i)\cdot P^{(3)}(\xi_j)\big)$*
- *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 $q \in \{0,1\}$:*

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

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

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

Proof. For the first item, Lemma 4.17 ensures that the pin words of $\oplus[\xi_i,\xi_j]$ encode pin representations reading both ξ_i and ξ_j in one time. 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 4.17, 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. □

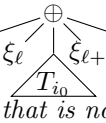
This completes the explicit description of all the sets of pin words appearing in Theorem 4.12.

4.3. Recursive case: decomposition trees with a linear root

In this section we 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 [7, Lemma 3.7] T_{i_0} is then the first child read by any pin representation. Lemma 4.6 (p.21) gives a characterization of permutations in which T_{i_0} may be read in several pieces. Moreover from Remark 4.7 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 4.6, the first part S to be read is not uniquely determined. For example every permutation satisfying $F3$ also satisfies $F1$, and some permutations satisfy both $F1$ and $G2$ (see Figure 11 p.23). In Figure 13 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 13. We say that a permutation satisfies condition (iHj) when its diagram has the corresponding shape in Figure 13 (up to symmetry) and does not satisfy any condition that appears above (iHj) in Figure 13. 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 4.6. Moreover as T_{i_0} is not an increasing oscillation, the sets S and T that appear on Figure 13 are such that $|S| \geq 2$ and $|T| \geq 1$.

Recall that $P(\pi)$ denotes the set of pin words encoding π. Let us also denote by $P(T)$ the set of pin words that encode the permutation whose decomposition tree is T.

Theorem 4.19. *Let* $\pi = \xi_1 \cdots \xi_\ell \; \xi_{\ell+2} \cdots \xi_r$  *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*

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

We describe below the set $P(\pi)$ *of pin words encoding* π*. When* $\pi \in \mathcal{H}$*, these sets are given when the diagram of* π *is one of those shown in Figure 13. When the diagram of* π *is one of their symmetries,* $P(\pi)$ *is modified accordingly.*

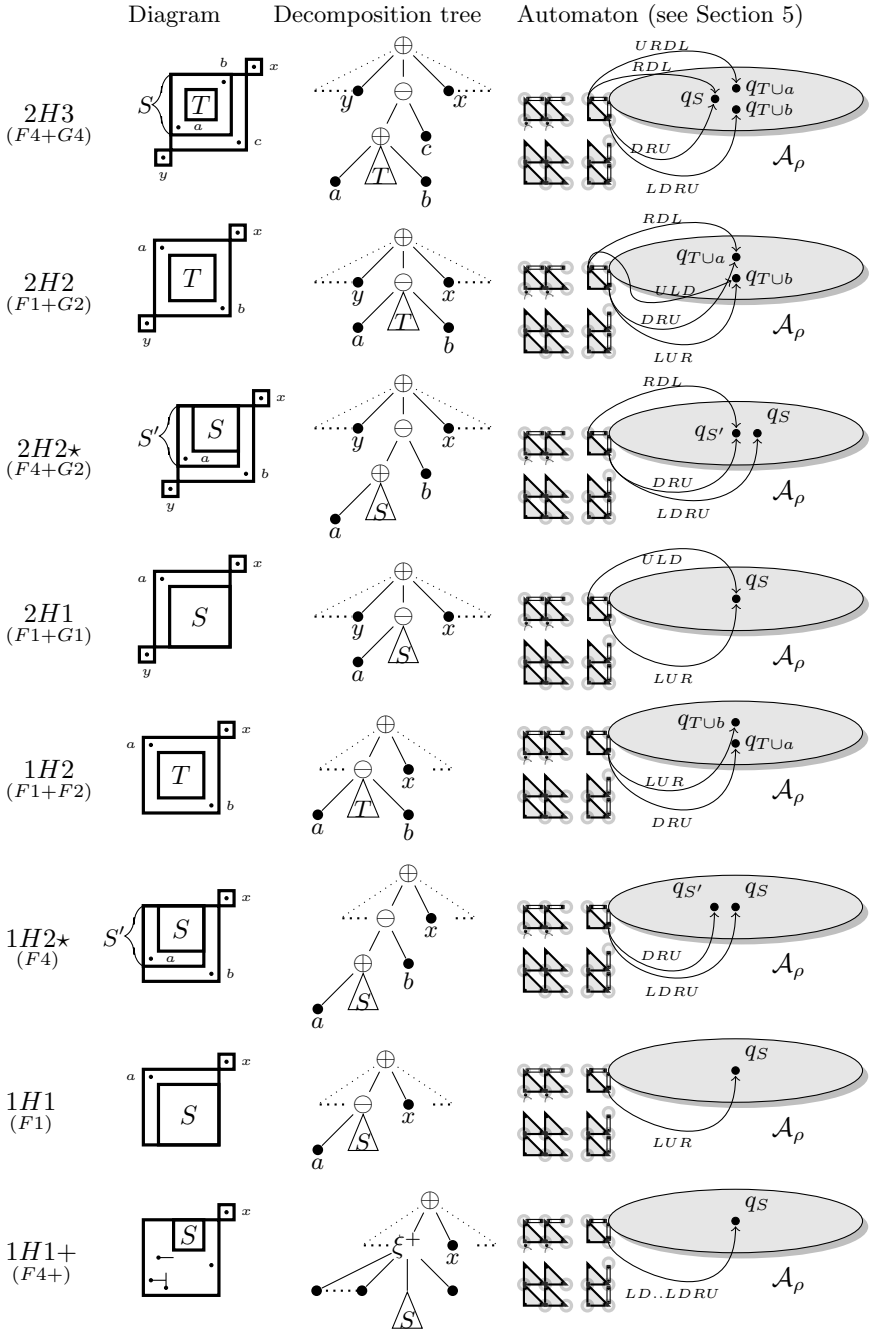


Figure 13: The set $\mathcal{H}$ and conditions (iHj): $\pi \in \mathcal{H}$ if and only if π satisfies one of the conditions (iHj) shown above up to symmetry, that form a partition of $\mathcal{H}$.

• *If* $\pi \notin \mathcal{H}$ *(*i.e.*, if* π *does not satisfy any condition shown up to symmetry on Figure 13) then* $P(\pi) = P_0 = P(T_{i_0}) \cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup \mathfrak{P}^{(3)}_{(\ell+2)}$.

• *If* π *satisfies condition* $(1H1)$ *then* $P(\pi) = P_0 \cup P_1$*, with*

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

• *If* π *satisfies condition* $(1H1+)$ *then* $P(\pi) = P_0 \cup P_1$*, with*

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

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 4.14 p.25) by deleting its first letter.*

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

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

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

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

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

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

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

$$P_1 = \underbrace{P(S) \cdot \underbrace{1}_{x} \cdot \underbrace{D}_{b} \cdot \underbrace{L}_{a}}_{x \bigcup T_{i_0}} \cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup \mathfrak{P}^{(3)}_{(\ell+3)}, \; P_2 = \underbrace{P(S') \cdot \underbrace{1}_{x} \cdot \underbrace{D}_{b}}_{x \bigcup T_{i_0}} \cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup \mathfrak{P}^{(3)}_{(\ell+3)}$$

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

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

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

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

- *If π satisfies condition $(2H3)$ then $P(\pi) = P_0 \cup P_1 \cup P_2 \cup P_3 \cup P_4$, with*

$$P_1 = \underbrace{P(S)\cdot \underbrace{1}_{x}\cdot\underbrace{D}_{c}}_{x\bigcup T_{i_0}}\cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup\!\sqcup \mathfrak{P}^{(3)}_{(\ell+3)},\ P_2 = \underbrace{P(T\cup b)\cdot \underbrace{1}_{x}\cdot\underbrace{D}_{c}\cdot\underbrace{L}_{a}}_{x\bigcup T_{i_0}}\cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup\!\sqcup \mathfrak{P}^{(3)}_{(\ell+3)},$$

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

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 [7, 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 4.19. In this case, by definition of $\mathcal{H}$, we know that T_{i_0} is read in one time. By Lemma 4.5 (p.20), the other children are also read in one time, and Lemma 4.3 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 the block.

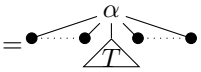
In the other cases of Theorem 4.19, by Lemma 4.6, every pin representation of π either reads T_{i_0} in one time or in two pieces. In case T_{i_0} is read in one time, the pin representation is as before encoded by pin words of P_0. If it is read in two pieces, Lemma 4.6 and its proof and Remark 4.7 ensure that the corresponding pin words are those described.

Consider for example π satisfying condition $(1H1)$. Then π satisfies condition $F1$ of Lemma 4.6, and only this one by definition of $(1H1)$. If T_{i_0} is read in two pieces, then Lemma 4.6 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 $S = T \cup a \cup b$. □

Remark 4.20. *If π is a $\ominus$-decomposable permutation, similar description of $P(\pi)$ can be obtained from Remark 4.1 (p.19).*

4.4. 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 $\pi =$ 

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 4.21. *For a permutation $\pi = \alpha[1, \dots, 1, T, 1, \dots, 1]$ with $\alpha = \alpha_1 \dots \alpha_k$, we define condition $(\mathcal{C})$ as follows:*

$(\mathcal{C})$ $\begin{cases} \bullet\ \alpha \text{ is an increasing – resp. decreasing – quasi-oscillation (see p.11);} \\ \bullet\ T \text{ expands an auxiliary point of } \alpha; \\ \bullet\ \text{the shape of } T \text{ is } \oplus\,T' \text{ – resp. } \ominus\,T' \text{ – if the auxiliary point is } \alpha_k \\ \text{or } \alpha_{k-1} \text{ and } \oplus\,T' \text{ – resp. } \ominus\,T' \text{ – if the auxiliary point is } \alpha_1 \text{ or } \alpha_2. \end{cases}$

Lemma 4.22. *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 time by p;*

(2) *$T = \{p_1, \dots, p_i\} \bigcup \{p_n\}$ with $i \neq n-1$, and π satisfies condition $(\mathcal{C})$;*

(3) *$p_1 \notin T$, $T = \{p_2, p_n\}$, and π satisfies condition $(\mathcal{C})$.*

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 14. If (2) is satisfied, defining T' as in condition $(\mathcal{C})$, 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 14 up to symmetry.

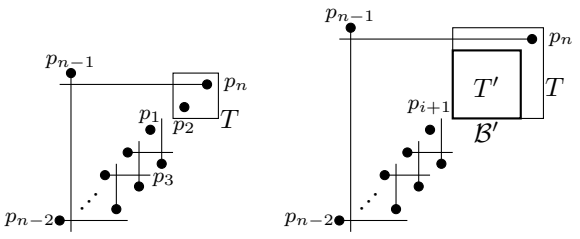


Figure 14: Diagram of π when one child T is not a leaf, is read in two pieces and $p_1 \notin T$ or $p_1 \in T$.

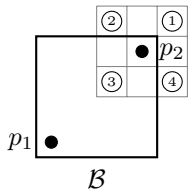
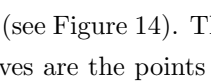


Figure 15: Possible positions for p_n.

Proof. If $p_1 \notin T$, then by Lemma 3.12*(ii)* of [7], $T = \{p_2, p_n\}$. 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 15. 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 2.1 (p.5). But $n > 3$ (α is simple so $|\alpha| \geq 4$) so that these positions are forbidden. Hence p_n lies in position ① and $T = 12$.

As α is a simple pin-permutation, p_3 respects the separability condition. But if p_3 lies above or on the right of the bounding box $\mathcal{B}$ 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 14). The same argument goes for every pin p_i with $i = 3, \ldots, n-2$ 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 $(\mathcal{C})$. 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 time. By Lemma 3.11*(i)* of [7], it is read in two pieces, the second part being p_n. Then $T = \{p_1, \ldots, p_i\} \bigcup \{p_n\}$ with $n \neq i+1$. Thus p_n does not lie on the sides of the bounding box $\mathcal{B}'$ of $\{p_1, \ldots, p_i\}$. Up to symmetry, we can assume that p_n lies in quadrant 1 with respect to $\mathcal{B}'$ (see Figure 14). Therefore, $T =$ 

, T' being the sub-forest of T whose leaves are the points in $\mathcal{B}'$, *i.e.*, are $p_1, \ldots, p_i$. As T is a block, no pin must lie on the sides of the bounding box of $\{p_1, \ldots, p_i, p_n\}$. Moreover p_{i+1} does not lie in quadrant 1 with respect to T, otherwise p_n would lie inside the bounding box of $\{p_1, \ldots, p_{i+1}\}$. If p_{i+1} lies in quadrant 2 or 4, p_n would lie on the side of the bounding box of $\{p_1, \ldots, p_{i+1}\}$ 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 $j \in \{i+1, \ldots, n-2\}$. 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 $(\mathcal{C})$ with $T' = \{p_1, \ldots, p_i\}$, as can be seen (up to symmetry) on the right part of Figure 14. Notice that given α and its auxiliary point, once we know that $T = \{p_1, \ldots, p_i\} \bigcup \{p_n\}$ with $i \neq n-1$ then $(p_{i+1}, \ldots, p_n)$ 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 time by p, concluding the proof. □

With the description of the pin representations of π in Lemma 4.22 and the definition that follows, we are able to give in Theorem 4.25 below an explicit description of the set $P(\pi)$ of pin words that encode π.

Definition 4.23. *For every simple pin-permutation α, with an active point x (see p.24) 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 4.24. *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 3 p.6 and Remark 2.4 p.6).*

Theorem 4.25. *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 $(\mathcal{C})$, then $P(\pi) = P(T) \cdot Q_x(\alpha)$.*
- *If π satisfies condition $(\mathcal{C})$, we define T' as in $(\mathcal{C})$, 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 4.22.

If p satisfies statement (1) of Lemma 4.22 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 $(\mathcal{C})$ or not.

If p satisfies statement (2) of Lemma 4.22 then π satisfies condition $(\mathcal{C})$. 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 4.22. 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 4.22, $T' = \{p_1, \dots, p_{k-1}\}$ with $k = |T|$ thus the prefix of length $k-1$ of u is in $P(T')$. From Lemma 4.22, $(p_k, \dots, p_n)$ is uniquely determined. Moreover, as $k \geq 3$, Remark 2.4 (p.6) 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 4.22 then π satisfies condition $(\mathcal{C})$, $|T| = 2$ 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 4.22. □

To make the set $P(\pi)$ of pin words in the statement of Theorem 4.25 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 4.25.

Remark 4.26. *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 2.1 (p.5), 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 2.4 (p.6).*

Lemma 4.27. *In Theorem 4.25, when π satisfies condition $(\mathcal{C})$ and $|T| \geq 3$, the word w is a strict pin word of size 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 $(\mathcal{C})$ and $|T| \geq 3$. Define T' as in condition $(\mathcal{C})$ 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 4.22, thus $T = \{p_1, \dots, p_i\} \cup \{p_n\}$ as in the second diagram of Figure 14 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 3.5 (p.15) $\phi^{-1}(w')$ 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 14. 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 4.24 to each pin representation of α whose first point read is x corresponds exactly one word of $Q_x(\alpha)$. Recall from Remark 4.26 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 oscillation and x is an auxiliary

point of α. By definition, every increasing oscillation has size at least 4, and we distinguish cases depending on whether it is of size greater than or at most 5.

Every increasing oscillation of size greater than 5 has exactly two active knights, with exactly one containing the auxiliary point (which is uniquely determined in this case). Indeed, from Lemma 4.6 of [7] (see also the last diagram of Figure 5 p.12), 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.

For increasing oscillations of size 4 or 5 (see Figure 6 p.13) we may also apply to Lemma 4.6 of [7] to count their active knights that involve the auxiliary point x. An increasing oscillation of size 5 has exactly 4 active knights, all of them contain the main substitution point, and exactly one of them contains the auxiliary point x. An increasing oscillation of size 4 has exactly 4 active knights and each of its point (including the auxiliary point x) belongs to exactly two active knights.

Consequently 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 6 (p.13). We then check by comprehensive verification (of the four cases of size 4 on Figure 6) that $\phi(\bar{w}')$ does not contain Z. Details are left to the reader. □

We are furthermore able to describe w explicitly in Remark 4.29 below, and we record its expression here for future use in our work.

Definition 4.28. *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

$w_\alpha^A = (DL)^{p-2}DRU$ *(resp.* $w_\alpha^A = (LD)^{p-2}LUR$*) if* $|\alpha| = 2p$

$w_\alpha^A = (DL)^{p-2}UR$ *(resp.* $w_\alpha^A = (LD)^{p-2}RU$*) if* $|\alpha| = 2p-1$*;*

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

$w_\alpha^A = (UR)^{p-2}ULD$ *(resp.* $w_\alpha^A = (RU)^{p-2}RDL$*) if* $|\alpha| = 2p$

$w_\alpha^A = (UR)^{p-2}DL$ *(resp.* $w_\alpha^A = (RU)^{p-2}LD$*) if* $|\alpha| = 2p-1$*;*

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 4.29. *If A is in the top right corner of α (see Figure 14 p.32 or Figure 5 p.12), 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 4.30. *In Theorem 4.25, if π satisfies condition $(\mathcal{C})$ and $|T| = 2$, then $P_{\{1,n\}}(\pi) \cup P_{\{2,n\}}(\pi) = \{u \in P(\pi) \mid u$ 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 2.4 p.6). Moreover, with the notations of Lemma 4.15 (p.26), and $K_{A,M}$ as in Definition 4.28, we have an explicit description of $P_{\text{SQS}}(\pi)$:*

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

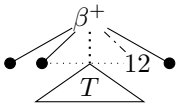
if α is increasing (see Figure 14 p.32), then

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

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 [7], 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 4.31. *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 of [7], the configuration depicted on Figure 16 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 2.4 (p.6). □

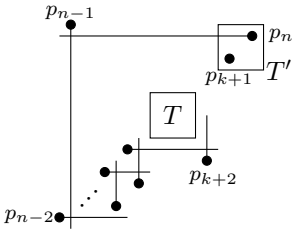


Figure 16: Diagram of π if two children are not leaves

Remark 4.32. *The word w in the statement of Theorem 4.31 is a strict pin word uniquely determined by β^+ and the two points expanded in β^+.*

More precisely, taking the notations of Definition 4.28 (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 16).

For decomposition trees whose root is a decreasing quasi-oscillation, we obtain a description of $P(\pi)$ similar to the one of Theorem 4.31. This is derived in the fashion explained in Remark 4.1 (p.19).

5. Building deterministic automata accepting the languages $\mathcal{L}_\pi$

Recall that our goal is, using Theorem 3.13 (p.18) and given a finite basis B, to find an algorithm checking if the language $\mathcal{M}\setminus\cup_{\pi\in B}\mathcal{L}_\pi$ is finite, $\mathcal{M}$ being the set of words of length at least 2 over $\{U,D,L,R\}$ such that L,R is followed by U,D and conversely (see p.14). For algorithmic reasons to be detailed later, we rather consider the language $\overleftarrow{\mathcal{M}\setminus\cup_{\pi\in B}\mathcal{L}_\pi}$ where for any language $\mathcal{L}$ we denote by $\overleftarrow{\mathcal{L}}$ the language $\{\overleftarrow{v}\mid v\in\mathcal{L}\}$ and for any word $v=v_1\dots v_p$ we denote by $\overleftarrow{v}=v_p\dots v_1$ the reverse of v. By definition of $\mathcal{M}$, we have $\overleftarrow{\mathcal{M}\setminus\cup_{\pi\in B}\mathcal{L}_\pi}=\overleftarrow{\mathcal{M}}\setminus\cup_{\pi\in B}\overleftarrow{\mathcal{L}_\pi}=\mathcal{M}\setminus\cup_{\pi\in B}\overleftarrow{\mathcal{L}_\pi}$. With usual constructions of automata theory (see [16] among other references), the construction of an automaton accepting the language $\mathcal{M}\setminus\cup_{\pi\in B}\overleftarrow{\mathcal{L}_\pi}$ is easily obtained from automata recognizing $\overleftarrow{\mathcal{L}_\pi}$ for $\pi\in B$.

From the definition of $\mathcal{L}_\pi$ given p.16 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 3.4 (p.14) 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.

In this section, we give for any pin-permutation π an explicit construction of a deterministic automaton $\mathcal{A}_\pi$ that recognizes the language $\overleftarrow{\mathcal{L}_\pi}$. We also present an alternative construction of $\mathcal{A}_\pi$ whose complexity is optimized; but instead of $\overleftarrow{\mathcal{L}_\pi}$, the automaton recognizes a language $\mathcal{L}$ such that $\mathcal{L}\cap\mathcal{M}=\overleftarrow{\mathcal{L}_\pi}\cap\mathcal{M}$. 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 $\mathcal{A}_\pi$ 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 $\overleftarrow{\phi(u^{(\ell)})}$ 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 $\mathcal{L}_\pi$ the factors $\overleftarrow{\phi(u^{(\ell)})}$ are separated by $A^\star$.

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 4.12 and 4.19 (p.24 and 28).

To avoid exponential complexity it is important that the construction provides deterministic automata. From deterministic automata $\mathcal{A}_\pi$ recognizing the languages $\overleftarrow{\mathcal{L}_\pi}$, we can obtain a deterministic automaton accepting words whose reverses encode proper pin-permutations containing some pattern $\pi\in B$, for any finite set B of patterns. This is achieved using the Cartesian product of the automata $\mathcal{A}_\pi$ 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 an automaton recognizing $\mathcal{M} \setminus \cup_{\pi \in B} \overleftarrow{\mathcal{L}_\pi}$), while the same operation on non-deterministic automata is exponential in the worst case. This last step will be detailed in Section 6.

For now, we focus on the construction of the automata $\mathcal{A}_\pi$. In Subsection 5.1, we present generic constructions of automata that will be used several times. In Subsections 5.2 to 5.5, we construct recursively the automata $\mathcal{A}_\pi$ 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.19. In these constructions, some states of the automata must be marked, and this is detailed in Subsection 5.6. We conclude with Subsection 5.7 that analyses the complexity of building $\mathcal{A}_\pi$.

5.1. 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 $A^\star X$ 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 5.1. *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 $A^\star X$). 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 $\mathcal{AC}^{\circlearrowleft}(X)$ that recognizes the set $A^\star X A^\star$ of words having a factor in X.

Remark 5.2. *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 5.1:

Lemma 5.3. *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 5.1. 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 5.1. □

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) [4], 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 $\mathcal{A}_1$ with 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.

[4] This 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.

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 $A^\star \cdot \mathcal{L}$ for an arbitrary language $\mathcal{L}$, we can do the same construction to obtain an automaton recognizing the concatenation $\mathcal{L}_1 \cdot \mathcal{L}_2 = \mathcal{L}_1 \cdot A^\star \cdot \mathcal{L}$. 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 $\mathcal{AC}^{\circlearrowleft}(X)$ and $\mathcal{A}$ is $\mathcal{AC}(X) \cdot \mathcal{A}$. Therefore, even though $\mathcal{AC}(X)$ recognizes a language strictly included in $A^\star X$, $\mathcal{AC}(X) \cdot \mathcal{A}$ recognizes $A^\star X A^\star \mathcal{L}$ when $\mathcal{A}$ recognizes a language $A^\star \mathcal{L}$.

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 complete[5]. 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 $(\mathcal{L}_1 \cup \mathcal{L}_2)A^\star = \mathcal{L}A^\star$ is recognized by the automaton $\mathcal{U}^{\circlearrowleft}(\mathcal{A}_1, \mathcal{A}_2)$ with an additional loop labeled by all letters of A on the final state. Notice that $\mathcal{U}(\mathcal{A}_1, \mathcal{A}_2)$ (resp. $\mathcal{U}^{\circlearrowleft}(\mathcal{A}_1, \mathcal{A}_2)$) is deterministic, almost complete (resp. complete) and has a unique final state without outgoing transitions (resp. with a loop labeled by all letters of A). The complexity in time and space of these constructions is in $\mathcal{O}(|\mathcal{A}_1| \cdot |\mathcal{A}_2|)$.

5.2. Pin-permutation of size 1 *and simple pin-permutations*

Pin-permutation of size 1.
When $\pi = 1$, we have $P(\pi) = \{1, 2, 3, 4\}$. Then

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

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 $\mathcal{A}_\pi$ of Figure 17.

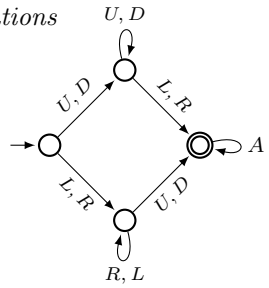


Figure 17: The automaton $\mathcal{A}_\pi$ when $\pi = 1$.

Simple pin-permutations. In this paragraph, for a simple permutation π whose set of pin words $P(\pi)$ is given, we build the automaton $\mathcal{A}_\pi$. The computation of

[5] Notice that the automata $\mathcal{AC}(X)$, $\mathcal{AC}^{\circlearrowleft}(X)$ and $\mathcal{AC}(X_1, X_2)$ satisfy these conditions.

$P(\pi)$ from π will be discussed in Subsection 6.2 as a subprocedure of the algorithm described in Section 6. The study that follows is based on the upcoming lemma.

Lemma 5.4. *For every permutation π (not necessarily simple), we have*

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

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

Proof. By definition of $\mathcal{L}(u)$ (see Definition 3.9 p.16),

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

Moreover, as can be seen on Figure 3 (p.6), 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

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

Hence

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

concluding the proof. □

Lemma 5.5. *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 $\mathcal{O}\left(|E_\pi^{\mathrm{S}}| \cdot |E_\pi^{\mathrm{QS}}| \cdot |\pi|^2\right)$ a deterministic complete automaton $\mathcal{A}_\pi^{\mathrm{SQS}}$ 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 5.4, we have

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

Recall that $\mathcal{AC}(\overleftarrow{E_\pi^{\mathrm{S}}})$, $\mathcal{AC}(\overleftarrow{E_\pi^{\mathrm{QS}}})$ and $\mathcal{AC}(\mathcal{M}_2)$ are automata recognizing respectively the set of words ending with a first occurrence of a word of $\overleftarrow{E_\pi^{\mathrm{S}}}$, $\overleftarrow{E_\pi^{\mathrm{QS}}}$ and $\mathcal{M}_2$ and obtained using the construction given in Section 5.1. The sizes of the first two automata are respectively $\mathcal{O}\left(|E_\pi^{\mathrm{S}}| \cdot |\pi|\right)$ and $\mathcal{O}\left(|E_\pi^{\mathrm{QS}}| \cdot |\pi|\right)$, and the

size of the third one is constant. Indeed for all w in E^{S}_π, $|w| = |\pi| + 1$ and for all w in E^{QS}_π, $|w| = |\pi|$ so that $\|E^{\mathrm{S}}_\pi\| = |E^{\mathrm{S}}_\pi| \cdot (|\pi| + 1)$ and $\|E^{\mathrm{QS}}_\pi\| = |E^{\mathrm{S}}_\pi| \cdot |\pi|$.

Then the deterministic automaton $\mathcal{A}^{\mathrm{SQS}}_\pi$ is obtained as the union $\mathcal{U}^{\circlearrowleft}(\mathcal{AC}(\overleftarrow{E^{\mathrm{S}}_\pi})\ ,\ \mathcal{AC}(\overleftarrow{E^{\mathrm{QS}}_\pi}) \cdot \mathcal{AC}(\mathcal{M}_2))$ in time and space $\mathcal{O}\left(|E^{\mathrm{S}}_\pi| \cdot |E^{\mathrm{QS}}_\pi| \cdot |\pi|^2\right)$. □

Lemma 5.5 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 5.6. *By Theorem 4.8 (p.23) 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 $\mathcal{A}_\pi = \mathcal{A}^{\mathrm{SQS}}_\pi$ (from Lemma 5.5) and the time and space complexity of the construction of $\mathcal{A}_\pi$ is quadratic w.r.t. $|\pi|$, as soon as the pin words of π are given.*

Notice also that when $\pi = 12$, $\mathcal{A}_\pi = \mathcal{A}^{\mathrm{SQS}}_\pi$ 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 $\mathcal{A}_\pi$ whose complexity is *linear* when the permutation π is simple. This construction relies on the same idea as the one we give in [6].

Lemma 5.7. *For every permutation π whose sets of strict and quasi-strict pin words (or equivalently E^{S}_π and E^{QS}_π) are given, one can build in time and space $\mathcal{O}\left((|E^{\mathrm{S}}_\pi| + |E^{\mathrm{QS}}_\pi|) \cdot |\pi|\right)$ a deterministic complete automaton $\mathcal{A}^{\mathrm{SQS}}_\pi$ 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 $E_\pi = E^{\mathrm{S}}_\pi \ \cup\ \mathcal{M}_2 \cdot E^{\mathrm{QS}}_\pi$ and $\mathcal{L} = A^\star \cdot \overleftarrow{E_\pi} \cdot A^\star$. 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 5.7, setting $\mathcal{A}^{\mathrm{SQS}}_\pi = \mathcal{AC}^{\circlearrowleft}(\overleftarrow{E_\pi})$.

From Lemma 5.4, we have

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

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

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

concluding the proof. □

Remark 5.8. *When π is a simple permutation, the automaton $\mathcal{A}_\pi^{\text{SQS}}$ of Lemma 5.7 recognizes a language $\mathcal{L}$ such that $\mathcal{L} \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$. In the optimized alternative construction of $\mathcal{A}_\pi$ mentioned at the beginning of Section 5, for a simple permutation π we take $\mathcal{A}_\pi = \mathcal{A}_\pi^{\text{SQS}}$ from Lemma 5.7 and the time and space complexity of building the automaton $\mathcal{A}_\pi$ is* linear *w.r.t. $|\pi|$ as soon as $P(\pi)$ is given.*

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

W.l.o.g. (see Remark 4.1 p.19), the only non-recursive case with a linear root is the one where $\pi = \oplus[\xi_1, \ldots, \xi_r]$, every ξ_i being an increasing oscillation. Theorem 4.12 (p.24) 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 sequence of pin words belonging to the shuffle product

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

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 4.13 (p.25), 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.38) is easily obtained and gives:

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

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

The automaton $\mathcal{A}_\pi$ 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 4.14 p.25), we can use the construction given in Section 5.1 to build the deterministic automata $\mathcal{A}(\xi_i, \xi_j) = \mathcal{AC}(\chi_i^{(1)}, \chi_j^{(3)})$. Figure 18 shows a diagram of $\mathcal{A}(\xi_i, \xi_j)$ and defines states s_{ij}, $f_{ij}^{(1)}$ and $f_{ij}^{(3)}$.

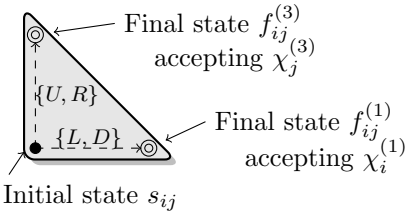


Figure 18: Atomic automaton $\mathcal{A}(\xi_i, \xi_j)$ used in the construction of $\mathcal{A}_\pi$.

Lemma 5.9. *For all i, j, the complexity in time and space of the construction of $\mathcal{A}(\xi_i, \xi_j)$ is $\mathcal{O}(|\xi_i| + |\xi_j|)$.*

Construction of $\mathcal{A}_{\xi_i}$. By Lemma 4.15 (p.26), 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 5.5 it is possible to build the deterministic automaton $\mathcal{A}_{\xi_i}$ in quadratic time and space w.r.t. $|\xi_i|$, and from Lemma 5.7 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 $P_{\text{SQS}} \uplus P(12) \cdot h$ where $h = 4$ (resp. 2) if $\xi_i = 231$ (resp. 312) and P_{SQS} is the set of strict and quasi-strict pin words of $P(\xi_i)$. With Lemma 5.5 or 5.7 we build the automaton $\mathcal{A}^{\text{SQS}}_{\xi_i}$ corresponding to P_{SQS}, 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.14) and $\mathcal{A}_{12}$ (see Remark 5.6 p.43). Finally the automaton $\mathcal{A}_{\xi_i}$ is the union $\mathcal{U}^{\circlearrowleft}(\mathcal{A}^{\text{SQS}}_{\xi_i}, \mathcal{AC}(\overleftarrow{\phi(h)}) \cdot \mathcal{A}_{12})$. As $|\xi_i| = 3$, $\mathcal{A}_{\xi_i}$ is built in constant time.

Lemma 5.10. *For all i, building $\mathcal{A}_{\xi_i}$ 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 4.18 (p.27) 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 $\mathcal{A}(\xi_i, \xi_{i+1})$, $\mathcal{A}_{\xi_i}$ and $\mathcal{A}_{\xi_{i+1}}$ as shown in Figure 19. More precisely $f^{(1)}_{i(i+1)}$ (resp. $f^{(3)}_{i(i+1)}$) and the initial state of $\mathcal{A}_{\xi_{i+1}}$ (resp. $\mathcal{A}_{\xi_i}$) are merged into a unique state that is neither initial nor final. The final states of $\mathcal{A}_{\xi_i}$ 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.

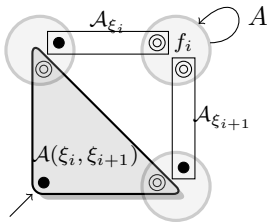


Figure 19: Automaton $\mathcal{A}^{\oplus}(\xi_i, \xi_{i+1})$.

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 5.6.

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_{SQS} and P', P_{SQS} 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 $\mathcal{A}^{\oplus}(\xi_i, \xi_{i+1}) = \mathcal{U}^{\circlearrowleft}\left(\mathcal{A}^{\text{SQS}}_{\oplus[\xi_i,\xi_{i+1}]}, \mathcal{AC}(\overleftarrow{\phi(3)}) \cdot \mathcal{A}_{\xi_{i+1}}\right)$. If $|\xi_{i+1}| = 3$, then $P' = P(\xi_{i+1}) \cdot 3 \bigcup P(12) \cdot 3X$ 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 5.11. *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 5.9 and the one of $\mathcal{A}_{\xi_i}$ is quadratic – or linear in the optimized

version – w.r.t. $|\xi_i|$ (see Lemma 5.10). From Lemma 4.18 (p.27), 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 $\mathcal{A}^{\text{SQS}}_{\oplus[\xi_i,\xi_{i+1}]}$ when $|\xi_i| = 1$ is quadratic – or linear in the optimized version – w.r.t. $|\xi_{i+1}|$, using Lemmas 5.5 and 5.7. 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. □

Assembling $\mathcal{A}_\pi$. According to the description of $\overleftarrow{\mathcal{L}_\pi}$ given p.44, 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 $\mathcal{A}_\pi$, as shown in Figure 20. More precisely for any i, j such that $1 \leq i < j \leq r$

- if $i + 1 \neq j$, then s_{ij}, $f^{(1)}_{(i-1)j}$ and $f^{(3)}_{i(j+1)}$ are merged into a unique state q_{ij} that is neither initial (except when $i = 1$ and $j = r$) nor final,
- if $i+1 = j$, $f^{(1)}_{(i-1)j}$, $f^{(3)}_{i(j+1)}$ 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. States q_{ij} defined above corresponds 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 $\left(A^\star\chi^{(1)}_1, \ldots, A^\star\chi^{(1)}_{i-1}\right) \sqcup\!\sqcup \left(A^\star\chi^{(3)}_r, \ldots, A^\star\chi^{(3)}_{j+1}\right) \cdot A^\star$ 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 $r = 2$, $\mathcal{A}_\pi$ is $\mathcal{A}^{\oplus}(\xi_1, \xi_2)$.

Lemma 5.12. *For any permutation π such that $\pi = \oplus[\xi_1, \ldots, \xi_r]$, every ξ_i being an increasing oscillation, the construction of the automaton $\mathcal{A}_\pi$ 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:

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

and from Lemmas 5.9 and 5.11, in the classical construction

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

and in the optimized version

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

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

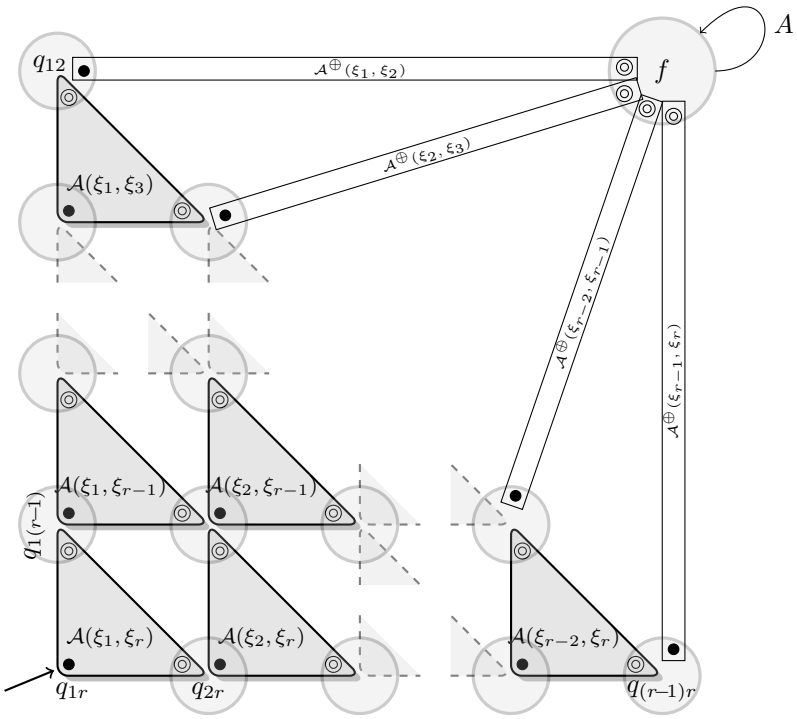


Figure 20: Automaton $\mathcal{A}_\pi$ for $\pi = \oplus[\xi_1, \ldots, \xi_r]$ where every ξ_i is an increasing oscillation.

Correctness of the construction. We now prove that the automaton $\mathcal{A}_\pi$ given in Figure 20 recognizes the language $\overleftarrow{\mathcal{L}_\pi}$, for $\pi = \oplus[\xi_1, \ldots, \xi_r]$, every ξ_i being an increasing oscillation.

Definition 5.13. *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 $A^\star$. We define $trace_{\mathcal{A}}(u)$ 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 $A^\star$. We define two parameters on u:

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

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

Remark 5.14. *Every word u such that $i(u) \geq i$ and $j(u) \leq j$ belongs to* $\Big((A^\star\chi_1^{(1)}, A^\star\chi_2^{(1)}, \ldots, A^\star\chi_i^{(1)}) \sqcup\!\sqcup (A^\star\chi_r^{(3)}, A^\star\chi_{r-1}^{(3)}, \ldots, A^\star\chi_j^{(3)})\Big) \cdot A^\star$.

Proof. By definition, u belongs to $A^\star{\cdot}\chi_1^{(1)}{\cdot}A^\star{\cdot}\chi_2^{(1)} \ldots A^\star{\cdot}\chi_i^{(1)}{\cdot}A^\star$ and to $A^\star{\cdot}\chi_r^{(3)}{\cdot} A^\star \cdot \chi_{r-1}^{(3)} \ldots A^\star \cdot \chi_j^{(3)} \cdot A^\star$. Moreover, by Remark 4.14 (p.25), 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 $\Big((A^\star\chi_1^{(1)}, A^\star\chi_2^{(1)}, \ldots, A^\star\chi_i^{(1)}) \sqcup\!\sqcup (A^\star\chi_r^{(3)}, A^\star\chi_{r-1}^{(3)}, \ldots, A^\star\chi_j^{(3)})\Big) \cdot A^\star$. □

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

Lemma 5.15. *Let Q be the set of states of $\mathcal{A}_\pi$ (see Figure 20), $(i,j) \neq (1,r)$ and $u \in A^\star$. Then $trace_{\mathcal{A}_\pi}(u) \in (Q \setminus q_{ij})^\star \cdot q_{ij}$ 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}. □

Theorem 5.16. *If $\pi = \oplus[\xi_1, \ldots, \xi_r]$ where every ξ_i is an increasing oscillation then automaton $\mathcal{A}_\pi$ given in Figure 20 recognizes the language $\overleftarrow{\mathcal{L}_\pi}$.*

Proof. Assume that $r \geq 3$ (otherwise $r = 2$, $\mathcal{A}_\pi$ is $\mathcal{A}^\oplus(\xi_1, \xi_2)$ and the result trivially holds). We first prove that every word recognized by $\mathcal{A}_\pi$ is in $\overleftarrow{\mathcal{L}_\pi}$. Let u be a word recognized by $\mathcal{A}_\pi$. Then $trace_{\mathcal{A}_\pi}(u)$ ends with the final state f of $\mathcal{A}_\pi$. As f is accessible only from some $\mathcal{A}^\oplus(\xi_k, \xi_{k+1})$, $trace_{\mathcal{A}_\pi}(u)$ 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 $trace_{\mathcal{A}_\pi}(u) \in q_{i_1j_1}Q^\star q_{i_2j_2}Q^\star \ldots q_{i_{r-1}j_{r-1}}Q^\star f$ 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.44, $u \in \overleftarrow{\mathcal{L}_\pi}$.

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 $\mathcal{A}_\pi$ and f its final state. The expression of $\overleftarrow{\mathcal{L}_\pi}$ given p.44 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 $v' = v_1 \ldots v_s$ 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 $v'' \in A^\star$ such that $v = v'v''$. So $w' = v''w$ belongs to $A^\star \cdot \overleftarrow{\mathcal{L}_{\oplus[\xi_k,\xi_{k+1}]}} = \overleftarrow{\mathcal{L}_{\oplus[\xi_k,\xi_{k+1}]}}$. Thus, using also Remark 5.14, we have:

$$u = \overbrace{\underbrace{\qquad\qquad}_{\in (A^\star\chi_1^{(1)}, \ldots, A^\star\chi_{k-1}^{(1)}) \sqcup\!\sqcup (A^\star\chi_r^{(3)}, \ldots, A^\star\chi_{k+2}^{(3)})}}^{v} \overbrace{\underbrace{\qquad\qquad}_{\in \overleftarrow{\mathcal{L}_{\oplus[\xi_k,\xi_{k+1}]}}}}^{w}$$

$$= \overbrace{\underbrace{\qquad\qquad}_{\in (A^\star\chi_1^{(1)}, \ldots, A^\star\chi_i^{(1)}) \sqcup\!\sqcup (A^\star\chi_r^{(3)}, \ldots, A^\star\chi_{i+3}^{(3)})}}^{v'} \overbrace{\qquad\qquad}^{w'}$$

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 $\bar{w} \in \chi^{(1)}_{i(v')}$, $i(\bar{v}) = i(v') - 1$ and $j(\bar{v}) = j(v')$; or $\bar{w} \in \chi^{(3)}_{j(v')}$, $i(\bar{v}) = i(v')$ and $j(\bar{v}) = j(v') + 1$. By Lemma 5.15, $trace_{\mathcal{A}_\pi}(v')$ 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 $u = v'w'$, $w' \in A^\star \cdot \chi^{(1)}_{i+1} \cdot A^\star \cdot \chi^{(1)}_{i+2} \cdot A^\star \dots A^\star \cdot \chi^{(1)}_{k-1} \cdot A^\star$. Therefore as $i \leq k-3$, $w' \in A^\star \cdot \chi^{(1)}_{i+1} \cdot A^\star \cdot \chi^{(1)}_{i+2} \cdot A^\star$ 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 $trace_{\mathcal{A}_\pi}(v')$ ends with $q_{i+1,i+2}$, $trace_{\mathcal{A}_\pi}(u)$ 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 $v'' \in A^\star \cdot \chi^{(1)}_{i+1} \cdot A^\star$. Moreover $w \in \overleftarrow{\mathcal{L}_{\oplus[\xi_{i+2},\xi_{i+3}]}}$ so $w' = v''w \in A^\star \cdot \chi^{(1)}_{i+1} \cdot \overleftarrow{\mathcal{L}_{\oplus[\xi_{i+2},\xi_{i+3}]}}$. Therefore $w' \in \overleftarrow{\mathcal{L}_{\oplus[\xi_{i+1},\xi_{i+2},\xi_{i+3}]}}$ and by Theorem 3.10(*ii*) (p.16), $w' \in \overleftarrow{\mathcal{L}_{\oplus[\xi_{i+1},\xi_{i+2}]}}$. Hence w' is recognized by $\mathcal{A}^{\oplus}(\xi_{i+1}, \xi_{i+2})$ so $q_{1r} \cdot u = f$ (since $q_{1r} \cdot v' = q_{i+1,i+2}$).

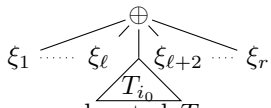
If $i = k$, by symmetry of $i(u)$ and $j(u)$ the proof is similar to the previous case, concluding the proof of Theorem 5.16. □

Remark 5.17. *With the optimized construction of $\mathcal{A}_\pi$, we prove similarly that $\mathcal{A}_\pi$ recognizes a language $\mathcal{L}$ such that $\mathcal{L} \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$.*

We end this section with a remark which will be useful in Subsection 5.6.

Remark 5.18. *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 20) as initial state and by considering only the states of $\mathcal{A}_\pi$ that are accessible from q_{2r}. Thus in $\mathcal{A}_\pi$ 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 19. 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 19.*

5.4. 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 $\mathcal{A}_\pi$ from $\mathcal{A}_\rho$.

If $\pi \notin \mathcal{H}$, i.e. *if π does not satisfy any condition of Figure 13 (p.29).* Then Theorem 4.19 (p.28) ensures that $P(\pi) = P(\rho) \cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup\!\sqcup \mathfrak{P}^{(3)}_{(\ell+2)}$ with

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

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

$$\overleftarrow{\mathcal{L}_\pi} = \Big(\Big(A^\star\chi^{(1)}_1, \dots, A^\star\chi^{(1)}_\ell\Big) \sqcup\!\sqcup \Big(A^\star\chi^{(3)}_r, \dots, A^\star\chi^{(3)}_{\ell+2}\Big)\Big) \cdot \overleftarrow{\mathcal{L}_\rho}$$

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}, $f^{(1)}_{ij}$ and $f^{(3)}_{ij}$ (see Figure 18 p.44). We furthermore introduce the deterministic automata $\mathcal{A}^{(1)}(\xi_i) = \mathcal{AC}(\chi^{(1)}_i)$ for $1 \le i \le \ell$ and $\mathcal{A}^{(3)}(\xi_j) = \mathcal{AC}(\chi^{(3)}_j)$ for $\ell + 2 \le j \le r$ whose initial and final states are denoted respectively $s^{(1)}_i$, $f^{(1)}_i$, $s^{(3)}_j$ and $f^{(3)}_j$. The automaton $\mathcal{A}^{(1)}(\xi_i)$ (resp. $\mathcal{A}^{(3)}(\xi_j)$) corresponds to the reading of parts of $\mathfrak{P}^{(1)}_{(\ell)}$ $\Big(\text{resp. } \mathfrak{P}^{(3)}_{(\ell+2)}\Big)$ in the shuffle product $\mathfrak{P}^{(1)}_{(\ell)} \sqcup\!\sqcup \mathfrak{P}^{(3)}_{(\ell+2)}$ after all the parts of $\mathfrak{P}^{(3)}_{(\ell+2)}$ $\Big(\text{resp. } \mathfrak{P}^{(1)}_{(\ell)}\Big)$ are read.

With these notations, the language $\overleftarrow{\mathcal{L}_\pi}$ associated to π is the one recognized by the automaton $\mathcal{A}_\pi$ given in Figure 21 where the following merges are performed:

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

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 $\mathcal{A}_\rho$ and the automata $\mathcal{A}^{(3)}(\xi_j)$ (resp. $\mathcal{A}^{(1)}(\xi_i)$) appear in $\mathcal{A}_\pi$ whose initial state is then q_r (resp. q_1).

The proof that the automaton $\mathcal{A}_\pi$ 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 5.19. *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 $\mathcal{A}_\pi$ (see Figure 21) is done in time and space $\mathcal{O}\big((|\pi| - |\rho|)^2\big)$ plus the additional complexity due to the construction of $\mathcal{A}_\rho$, both in the classical and the optimized construction.*

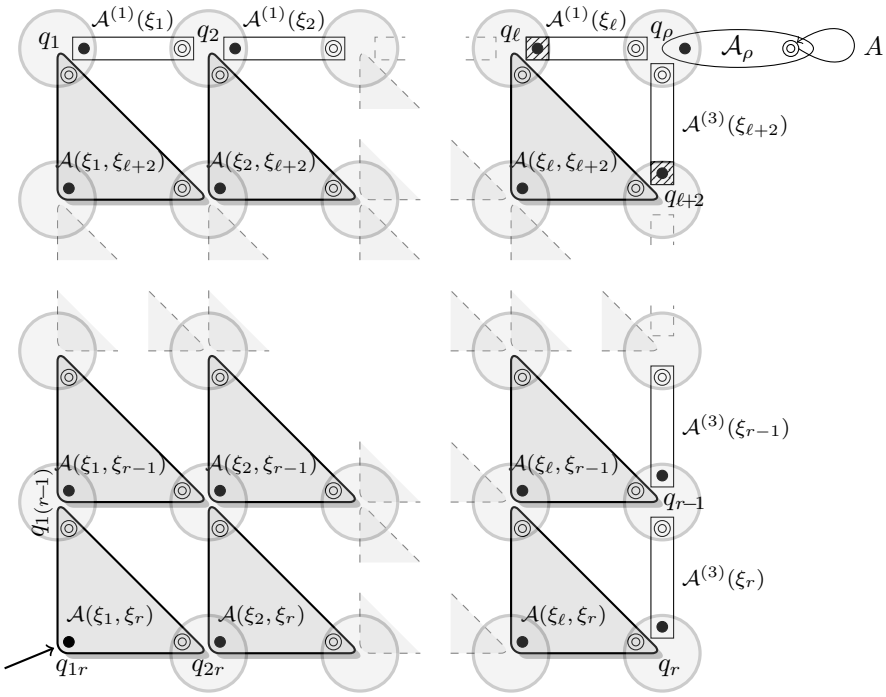


Figure 21: The automaton $\mathcal{A}_\pi$ for $\pi = \oplus[\xi_1, \ldots, \xi_\ell, \rho, \xi_{\ell+2}, \ldots, \xi_r]$, where every ξ_i but ρ is an increasing oscillation (in the case $\pi \notin \mathcal{H}$).

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 5.9 (p.44) and the fact that $|P^{(1)}(\xi_i)| \leq 2$ and $|P^{(3)}(\xi_j)| \leq 2$ (see Remark 4.14 p.25), 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. □

We end this paragraph with a remark which will be useful in Subsection 5.6.

Remark 5.20. *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 21) as initial state and by considering only the states of $\mathcal{A}_\pi$ that are accessible from q_{2r}. Thus in $\mathcal{A}_\pi$ 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 13 (p.29) holds for π.* Then Theorem 4.19 (p.28) ensures that $P(\pi)$ is the union of the set $P_0 = P(\rho) \cdot \mathfrak{P}^{(1)}_{(\ell)} \sqcup\!\sqcup \mathfrak{P}^{(3)}_{(\ell+2)}$ 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 $\mathcal{A}_\pi$ recognizes reversed words, these similar ends lead to similar beginnings in the automaton. So the automaton $\mathcal{A}_\pi$ has the same general structure as automaton $\mathcal{A}$ of Figure 21 but some transitions are added to account for the words in $P(\pi)$ not belonging to P_0.

More precisely $\mathcal{A}_\pi$ is obtained performing the following modifications on the automaton $\mathcal{A}$ of Figure 21. First we add new paths as depicted in the last column of Figure 13 (p.29). These paths start in the shaded states q_ℓ or $q_{\ell+2}$ of Figure 21 and arrive in marked states of $\mathcal{A}_\rho$. If a path is labeled in Figure 13 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 states of $\mathcal{A}_\rho$. These marked states may be seen as initial states of subautomata: in Figure 13, for all Y, q_Y is a state of $\mathcal{A}_\rho$ such that the language recognized from q_Y is $\overleftarrow{\mathcal{L}_\sigma}$, where σ is the permutation whose diagram is Y. The way such states of $\mathcal{A}_\rho$ are marked is explained in Section 5.6.

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 $\mathcal{A}^{(1)}(\xi_\ell) = \mathcal{AC}(\chi^{(1)}_\ell)$ (resp. $\mathcal{A}^{(3)}(\xi_{\ell+2}) = \mathcal{AC}(\chi^{(3)}_{\ell+2})$). Remark 4.14 (p.25) ensures that $\chi^{(1)}_\ell = \overleftarrow{\phi(P^{(1)}(\xi_\ell))}$ (resp. $\chi^{(3)}_{\ell+2} = \overleftarrow{\phi(P^{(3)}(\xi_{\ell+2}))}$) is defined on the alphabet $\{L, D\}$ (resp. $\{U, R\}$). Therefore, from Lemma 5.3 (p.40), 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 13, 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 $\mathcal{A}_\pi$. 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 13). 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 5.3 (p.40), 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. $\mathcal{A}^{(3)}(\xi_{\ell+2})$).

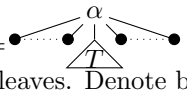
The proof that the automaton $\mathcal{A}_\pi$ obtained by this construction recognizes $\overleftarrow{\mathcal{L}_\pi}$ is omitted to avoid the examination of the eight cases of Figure 13. However, it is similar to the proof of Theorem 5.23 (p.55), with some of the difficulties released (since labels on the new paths are explicit in Figure 13, while they are not in the proof of Theorem 5.23).

Lemma 5.21. *The complexity of building $\mathcal{A}_\pi$ given in Lemma 5.19 (p.50) still holds if $\pi \in \mathcal{H}$.*

Proof. When $\pi \in \mathcal{H}$, the construction of $\mathcal{A}_\pi$ 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 $\mathcal{A}_\pi$ 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 $\mathcal{A}_\rho$ is bigger than $\mathcal{O}(|\rho|)$, this does not change the overall estimation of the complexity of the construction of $\mathcal{A}_\pi$ given in Lemma 5.19. □

5.5. Pin-permutations with a prime root: recursive case

5.5.1. Exactly one child of the root is not a leaf.

Let $\pi =$ 

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 4.23 p.33) 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 ($\mathcal{C}$) (see Definition 4.21 p.32). Then from Theorem 4.25 (p.34), $P(\pi) = P(\rho) \cdot Q_x(\alpha)$. 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 $\mathcal{A}_\pi$ recognizing $\overleftarrow{\mathcal{L}_\pi}$ is obtained by the concatenation of $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ with $\mathcal{A}_\rho$, which is recursively obtained.

If π satisfies condition ($\mathcal{C}$) and $|T| \geq 3$. Then by Theorem 4.25 (p.34) – 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 ($\mathcal{C}$), 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 4.27 (p.35) 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 $\mathcal{A}_\pi$ is the concatenation of the automaton $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ with $\mathcal{A}_\rho$ 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 4.27 ensures that no word of $\overleftarrow{\phi(Q_x(\alpha))}$ contains Z and therefore by Lemma 5.3 (p.40) all the transitions labeled by Z in $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ go to q_0. We built an automaton $\mathcal{A}$ by performing the following modifications on $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$: 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 $\mathcal{A}_\pi$ is then obtained from $\mathcal{A}$ and $\mathcal{A}_\rho$ by merging f with the initial state q_T of $\mathcal{A}_\rho$ and f' with a marked state $q_{T'}$ (see Section 5.6) of $\mathcal{A}_\rho$ which is a state from which the recognized language is $\overleftarrow{\mathcal{L}_{\rho'}}$. This construction is shown in Figure 22.

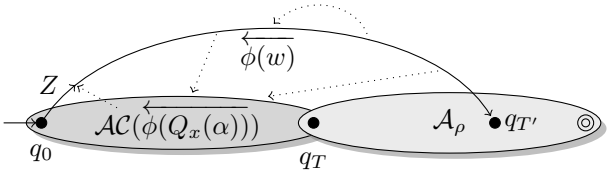


Figure 22: Automaton $\mathcal{A}_\pi$ for $\pi = \alpha[1, \ldots, 1, \rho, 1, \ldots, 1]$.

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 4.27), 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 5.22. *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 4.27 (p.35), 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 5.3 (p.40) 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. □

Lemma 5.22 allows us to prove the correctness of the above construction of $\mathcal{A}_\pi$. The idea is the following: if $u \in A^\star \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}_\rho}$ (resp. $A^\star \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}}$) and if $trace_{\mathcal{A}_\pi}(u)$ contains $q_{T'}$ (resp. q_T) and not q_T (resp. $q_{T'}$) before, then u is still accepted by $\mathcal{A}_\pi$ 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 5.23. *The automaton $\mathcal{A}_\pi$ obtained by the above construction recognizes $\overleftarrow{\mathcal{L}_\pi}$.*

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

Let u be a word of $\overleftarrow{\mathcal{L}_\pi}$. From Lemma 4.27 (p.35), 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 5.22 $q_0 \cdot u_1$ 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 $\overleftarrow{\mathcal{L}_\rho} = A^\star \cdot \overleftarrow{\mathcal{L}_\rho}$ (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 3.10 (p.16) 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 $\mathcal{A}_\pi$. 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 $u \notin A^\star \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}}$. Then since $u \in \overleftarrow{\mathcal{L}_\pi}$, we have $u \in A^\star \cdot \overleftarrow{\phi(Q_x(\alpha))} \cdot \overleftarrow{\mathcal{L}_\rho}$. Because u_1 ends with the first occurrence of a factor of $\overleftarrow{\phi(Q_x(\alpha))}$, we deduce that $u_2 \in A^\star \cdot \overleftarrow{\mathcal{L}_\rho} = \overleftarrow{\mathcal{L}_\rho}$.

Otherwise $u \in A^\star \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}}$. 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 5.22 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 4.27 (p.35) 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, $u_2 \in A^\star \cdot \overleftarrow{\phi(w)} \cdot \overleftarrow{\mathcal{L}_{\rho'}}$. Let $v \in A^\star$ and $v' \in \overleftarrow{\mathcal{L}_{\rho'}}$ such that $u_2 = v \cdot \overleftarrow{\phi(w)} \cdot v'$. From Lemma 4.27 p.35, 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 $u_2 = v \cdot \overleftarrow{\phi(w)} \cdot v' \in A^\star \cdot \overleftarrow{w'} \cdot A^\star \cdot \overleftarrow{\mathcal{L}_{\rho'}} \subseteq \overleftarrow{\mathcal{L}_\rho}$, concluding the proof. □

Remark 5.24. *With the optimized construction of $\mathcal{A}_\pi$, we prove similarly that $\mathcal{A}_\pi$ recognize a language $\mathcal{L}$ such that $\mathcal{L} \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$.*

If π satisfies condition ($\mathcal{C}$) and $|T| = 2$. Then the construction is no more recursive. Permutation π and its pin words are explicit. More precisely from Theorem 4.25 (p.34), $P(\pi) = P_{\{1,n\}}(\pi) \cup P_{\{2,n\}}(\pi) \cup P(T) \cdot Q_x(\alpha)$. Thus from Remark 4.30 (p.37),

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

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

Lemma 5.25. *Let $\pi = \alpha[1, \ldots, 1, \rho, 1, \ldots, 1]$ where α is a simple permutation whose set $P(\alpha)$ of pin words is given. Then the construction of the automaton $\mathcal{A}_\pi$ is done in time and space $\mathcal{O}(|\pi| - |\rho|)$ plus the additional time and space due to the construction of $\mathcal{A}_\rho$, except when π satisfies condition ($\mathcal{C}$) and $|T| = 2$. In this latter case, the complexity is $\mathcal{O}\left(|\pi|^3\right)$ with the classical construction and $\mathcal{O}\left(|\pi|^2\right)$ in the optimized version.*

Proof. Recall that $Q_x(\alpha)$ contains words of size $|\alpha| - 1$. Its cardinality is smaller than the one of $P(\alpha)$, hence smaller than 48 (see Theorem 4.8 p.23). Moreover $Q_x(\alpha)$ can be determined in linear time w.r.t. $|\alpha|$ as described in Remark 4.26 (p.35). Consequently, $\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})$ is built in time and space $\mathcal{O}(|\alpha|) = \mathcal{O}(|\pi| - |\rho|)$.

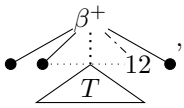
If π does not satisfy condition ($\mathcal{C}$) then $\mathcal{A}_\pi = \mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho$, so that $|\mathcal{A}_\pi| = |\mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))})| + |\mathcal{A}_\rho|$ and the time complexity of this construction is $\mathcal{O}(|\pi|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

If π satisfies condition ($\mathcal{C}$) and $|T| \geq 3$, then $|w| = |\alpha|$ and by Remark 4.29 (p.36), w is explicitly determined. Consequently, so is the additional path labeled by $\overleftarrow{\phi(w)}$ added to the automaton (see Figure 22). 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 $\mathcal{A}_\pi$ is built in $\mathcal{O}(|\pi| - |\rho|)$ time and space plus the additional time and space to build $\mathcal{A}_\rho$.

If π satisfies condition $(\mathcal{C})$ and $|T| = 2$, then $\mathcal{A}_\pi = \mathcal{U}^{\circlearrowleft}(\mathcal{A}_\pi^{\text{SQS}}, \mathcal{AC}(\overleftarrow{\phi(Q_x(\alpha))}) \cdot \mathcal{A}_\rho)$. Recall that $P_{\text{SQS}}(\pi)$ is given in Remark 4.30 (p.37) and contains 12 pin words. Hence, with the classical construction (resp. in the optimized version), from Lemma 5.5 (p.42) (resp. Lemma 5.7 p.43) and Remark 4.30, we can build $\mathcal{A}_\pi^{\text{SQS}}$ in time and space $\mathcal{O}\left(|\pi|^2\right)$ (resp. $\mathcal{O}(|\pi|)$). Moreover since $|\rho| = 2$, $\mathcal{A}_\rho$ 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, $\mathcal{A}_\pi$ is built in time and space $\mathcal{O}\left(|\pi|^3\right)$ (resp. $\mathcal{O}\left(|\pi|^2\right)$) with the classical (resp. optimized) construction. □

5.5.2. 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 4.31 (p.37) 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 size $|\beta^+|$ uniquely determined by β^+ and the two points expanded in β^+, and known explicitly from Remark 4.32 (p.37).

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

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

5.6. Marking states

In our constructions of Subsections 5.4 and 5.5 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 $\mathcal{A}_\pi$ in Section 5.4 for a permutation π whose decomposition tree has a linear root and satisfies a condition (iHj) of Figure 13 (p.29), or in Section 5.5 for a permutation π whose decomposition tree has a prime root and satisfies condition $(\mathcal{C})$ (see Definition 4.21 p.32) with $|T| \geq 3$.

In both cases we need to mark in the subautomaton $\mathcal{A}_\rho$ 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 13 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 $R =$ 

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 $\mathcal{A}_\theta$ the state q such that when starting from q the language recognized is the one recognized by $\mathcal{A}_\gamma$ (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 5.5).

The automaton $\mathcal{A}_\theta$ is obtained as described in Subsection 5.5, 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 $\mathcal{A}_\theta$ 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 $\mathcal{A}_\theta$ is the concatenation of two automata the second of which is $\mathcal{A}_\gamma$ whose initial state can be readily marked.

If ξ^+ has size 4, then $\xi^+ = 2\,4\,1\,3$ or $3\,1\,4\,2$ is a quasi-oscillation and θ may satisfy condition $(\mathcal{C})$. If it is not the case, $\mathcal{A}_\theta$ is obtained as above and so is the marking of state q. If on the contrary θ satisfies condition $(\mathcal{C})$, the construction of $\mathcal{A}_\theta$ depends on $|S|$. If $|S| \geq 3$, $\mathcal{A}_\theta$ is again the concatenation of two automata the second one being $\mathcal{A}_\gamma$, but with some states and transitions added. As these transitions are not reachable from the initial state of $\mathcal{A}_\gamma$, 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 $\mathcal{A}_\theta$ a state q corresponding to the initial state of $\mathcal{A}_\gamma$. But in the construction of $\mathcal{A}_\theta$ in Subsection 5.5, we have built an automaton $\mathcal{A}'$ such that $\mathcal{A}_\theta = \mathcal{U}^{\circlearrowleft}(\mathcal{A}_\theta^{\text{SQS}}, \mathcal{A}' \cdot \mathcal{A}_\gamma)$. Therefore $\mathcal{A}_\theta$ is a Cartesian product and the state q has been replicated several times. As $|S| = 2$, $\mathcal{A}_\gamma$ has a constant size, hence in this particular case we just duplicate it and mark its initial state instead of marking a state inside $\mathcal{A}_\theta$.

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 $\mathcal{A}_\rho$ 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 5.18 and 5.20 (p.49 and 51), the corresponding marked states of $\mathcal{A}_\rho$ (which is built as described in Section 5.3 or 5.4) are $q_{1(r-1)}$ and q_{2r} in Figure 20 (p.47) or 21 (p.51) – with

ρ instead of π. There is however one exception, corresponding to the special case described in Remark 5.18: when R has exactly two children, which are z and an increasing oscillations ξ. In this special case the construction of $\mathcal{A}_\rho$ is not recursive anymore. Instead of marking in $\mathcal{A}_\rho$ a state q corresponding to the initial state of $\mathcal{A}_\xi$, we just duplicate $\mathcal{A}_\xi$ 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 $\mathcal{A}_\pi$.

The few cases where the marked stated 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 except that we have to mark states in $\mathcal{A}_{\rho'}$ instead of $\mathcal{A}_\rho$, where ρ' is the permutation whose decomposition tree is R' and $\mathcal{A}_{\rho'}$ is a subautomaton of $\mathcal{A}_\rho$ 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 $(2H3)$, $(2H2\star)$ and $(1H2\star)$ where two levels of recursion are involved.

5.7. Complexity analysis

Theorem 5.26. *For every pin-permutation π of size n, $\mathcal{A}_\pi$ 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 $\mathcal{A}_\pi$, we first need to decide which shape of Equation $(\star)$ 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 6.2 as a subprocedure of the global algorithm of Section 6.

Then Theorem 5.26 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.

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 $\mathcal{A}_\rho$ 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 $\mathcal{A}_\pi$ can be built in time $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(n^4)$), n being the size of $|\pi|$.

pin-permutation of size n	Complexity	Optimized	Lemma
size 1	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
simple	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	5.6, 5.8
root $\oplus$ non-recursive	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	5.12
root $\oplus$ recursive, one child T is not an increasing oscillation	$\mathcal{O}(\,(n-\lvert\rho\rvert)^2\,)$ + time for $\mathcal{A}_\rho$	$\mathcal{O}(\,(n-\lvert\rho\rvert)^2\,)$ + time for $\mathcal{A}_\rho$	5.19, 5.21
root is prime recursive, $\mathcal{C}$ is satisfied, and T has size 2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	5.25
root is prime recursive (if not preceding case)	$\mathcal{O}(\,n-\lvert\rho\rvert\,)$ + time for $\mathcal{A}_\rho$	$\mathcal{O}(\,n-\lvert\rho\rvert\,)$ + time for $\mathcal{A}_\rho$	5.25, §5.5.2

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

Indeed let K be the number of levels of recursion needed in the construction of $\mathcal{A}_\pi$. 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 $\mathcal{A}_\pi$ is:

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

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 $\mathcal{A}_\pi$ is:

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

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

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

In this section, we put together the steps of the algorithm determining whether a permutation class $\mathcal{C} = Av(B)$ contains a finite number of simple permutations, under the condition that B is finite and given in input. Denoting k the number of pin-permutations in B, $s = \max\{|\pi| : \pi \text{ pin-permutation} \in B\}$, and $n = \sum_{\pi \in B} |\pi|$, the complexity of the algorithm is polynomial w.r.t. n and s and exponential w.r.t. k. The algorithm can be decomposed into several steps.

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

Following [12] (see Theorem 2.7 p.6) we first check if $\mathcal{C}$ contains finitely many parallel alternations and wedge simple permutations. From Lemmas 2.8, 2.9 and 2.10 (p.7) this problem is equivalent to testing if permutations of B contain some patterns of size at most 4. Using a result of [2], this can be done in $\mathcal{O}(n \log n)$ time (see Lemma 2.11 p.7).

6.2. 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 [7].

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 [8] or [**?**], and next the labels of linear and prime nodes as explained in [**?** , §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 $\mathcal{A}_\pi$ (see Section 5) 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 $2\,4\,1\,6\,3\,8\,5\ldots$ (see Figure 5 p.12) 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 [7] (see Equation ($\star$) p.19).

- 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 [6] we compute the set of pin words of α and test its emptiness. By Lemma 4.1 of [6], 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. From Remark 4.24 (p.33) we just have to test the emptiness of $Q_x(\alpha)$, which is computed in linear time w.r.t. $|\alpha|$ (see Remark 4.26 p.35). 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 Section 4.

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

From Theorem 3.13 (p.18) 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 (which we assume has been performed), 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 $\mathcal{L}_\pi$ is empty when π is not a pin-permutation (see p.16). We build the automaton $\mathcal{A}_\mathcal{C}$ as follows.

• First for each pin-permutation $\pi \in PB$, we construct $\mathcal{A}_\pi$ – which is deterministic and complete – recognizing a language $\mathcal{L}'_\pi$ such that $\mathcal{L}'_\pi \cap \mathcal{M} = \overleftarrow{\mathcal{L}_\pi} \cap \mathcal{M}$. This construction is performed in time and space at most $\mathcal{O}(|\pi|^2)$ as described in Section 5, with the optimized construction (see Theorem 5.26 p.59). Notice that the construction of $\mathcal{A}_\pi$ 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 $\mathcal{L}'_\pi$ 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 $\mathcal{A}_\pi$. 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 23 in time and space $\mathcal{O}(|\mathcal{A}_1|.|\mathcal{A}(\mathcal{M}))|) = \mathcal{O}(\prod_{\pi\in PB}|\pi|^2)$.

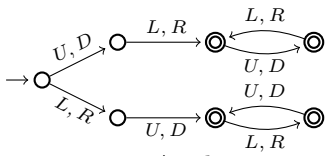


Figure 23: A deterministic automaton $\mathcal{A}(\mathcal{M})$ recognizing the set $\mathcal{M}$ of words of length at least 2 without any factor in $\{UU, UD, DU, DD, RR, RL, LR, LL\}$.

The automaton $\mathcal{A}_2$ recognizes $\left(\bigcup_{\pi\in PB}\mathcal{L}'_\pi\right)\cap \mathcal{M} = \left(\bigcup_{\pi\in PB}\overleftarrow{\mathcal{L}_\pi}\right)\cap\mathcal{M}$. By Lemma 3.12 (p.17) 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 2.6 (p.6) such permutations are necessarily proper pin-permutations.

• Next we complement $\mathcal{A}_2$ to build a deterministic automaton $\mathcal{A}_3$ recognizing $A^\star \setminus \left(\left(\bigcup_{\pi\in PB}\overleftarrow{\mathcal{L}_\pi}\right)\cap\mathcal{M}\right)$. 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 $\mathcal{O}(|\mathcal{A}_3|.|\mathcal{A}(\mathcal{M}))|) = \mathcal{O}(\prod_{\pi\in PB}|\pi|^2)$. The automaton $\mathcal{A}_\mathcal{C}$ recognizes $\mathcal{M}\setminus\left(\bigcup_{\pi\in PB}\overleftarrow{\mathcal{L}_\pi}\right)$. 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 3.13 (p.18), 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 com-

plexity 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.

6.4. Main result

We are now able to state the main theorem of this article:

Theorem 6.1. *Given a finite set of permutations B, we have described an algorithm that determines whether the permutation class $\mathcal{C} = Av(B)$ 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 $\mathcal{O}(n \log n + p^2)$.*

Proof. We determine whether there are finitely many parallel alternations and wedge simple permutations in $\mathcal{C}$ in $\mathcal{O}(n \log n)$ time. Then we compute the set PB of pin-permutations of B and their decomposition trees with some additional information (like the set of pin words that encode each simple permutation labeling a node) in $\mathcal{O}(n)$ time. Finally we check whether there are finitely many proper pin-permutations in $\mathcal{C}$ as explained above (*i.e.*, by looking for a cycle in the automaton $\mathcal{A}'_\mathcal{C}$) in $\mathcal{O}(p^2)$ time which is $\mathcal{O}(s^{2k})$. □

7. Conclusion

The work reported here follows the line opened by [3] and continued by [12]. 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, [12] 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 [7], 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 [14], 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 method to do so is already described in [3], but it is of highly exponential complexity. An algorithm for this second step has subsequently been described in [**?**], 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 [**?**] 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 [14] 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 Boltzmann

random samplers of permutations in a class, or to the automatic evaluation of the Stanley-Wilf growth rate of a class.

References

[1] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[2] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366, London, UK, 2001. Springer-Verlag.

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

[4] Michael H. Albert, Mike D. Atkinson, Mathilde Bouvel, Nik Ruškuc, and Vincent Vatter. Geometric grid classes of permutations. *Transactions of the American Mathematical Society*, To appear, 2013.

[5] Michael H. Albert, Mike D. Atkinson, Robert Brignall, Nik Ruškuc, Rebecca Smith, and Julian West. Growth rates for subclasses of Av(321). *Electronic Journal of Combinatorics*, 17:Paper R141, 2010.

[6] Michael H. Albert, Mike D. Atkinson, and Martin Klazar. The enumeration of simple permutations. *Journal of Integer Sequences*, 6, 2003.

[7] Michael H. Albert, Steve Linton, and Nik Ruškuc. The insertion encoding of permutations. *Electronic Journal of Combinatorics*, 12:Paper R47, 2005.

[8] Mike D. Atkinson, Nik Ruškuc, and Rebecca Smith. Substitution-closed pattern classes. *Journal of Combinatorial Theory, Series A*, 118(2):317–340, 2011.

[9] Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, Carine Pivoteau, and Dominique Rossin. Combinatorial specification of permutation classes. In *Proceedings of FPSAC 2012 (24th International Conference on Formal Power Series and Algebraic Combinatorics), Nagoya, Japan*, volume AR of *DMTCS proceedings*, pages 781–792, 2012.

[10] Frédérique Bassino, Mathilde Bouvel, Adeline Pierrot, and Dominique Rossin. Deciding the finitness of the number of simple permutations contained in a wreath-closed class is polynomial. *Pure Mathematics and Applications*, 21(2):119–135, 2010.

[11] Frédérique Bassino, Mathilde Bouvel, and Dominique Rossin. Enumeration of pin-permutations. *Electronic Journal of Combinatorics*, 18(1):Paper P57, 2011.

[12] Anne Bergeron, Cédric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. *SIAM Journal on Discrete Mathematics*, 22(3):1022–1039, June 2008.

[13] Mireille Bousquet-Mélou. Four classes of pattern-avoiding permutations under one roof: Generating trees with two labels. *Electronic Journal of Combinatorics*, 9(2):Paper R19, 2002.

[14] Mathilde Bouvel, Cédric Chauve, Marni Mishna, and Dominique Rossin. Average-case analysis of perfect sorting by reversals. *Discrete Mathematics, Algorithms and Applications*, 3(3):369–392, 2011.

[15] Robert Brignall. A survey of simple permutations. In Steve Linton, Nik Ruškuc, and Vincent Vatter, editors, *Permutation Patterns, St Andrews 2007*, volume 376 of *London Mathematical Society Lecture Note Series*, pages 41–65. Cambridge University Press, Cambridge, 2010.

[16] Robert Brignall. Grid classes and partial well order. *Journal of Combinatorial Theory, Series A*, 119:99–116, 2012.

[17] Robert Brignall, Sophie Huczynska, and Vincent Vatter. Decomposing simple permutations, with enumerative consequences. *Combinatorica*, 28(4):385–400, 2008.

[18] Robert Brignall, Sophie Huczynska, and Vincent Vatter. Simple permutations and algebraic generating functions. *Journal of Combinatorial Theory, Series A*, 115(3):423–441, 2008.

[19] Robert Brignall, Nik Ruškuc, and Vincent Vatter. Simple permutations: decidability and unavoidable substructures. *Theoretical Computer Science*, 391(1-2):150–163, 2008.

[20] Binh-Minh Bui Xuan, Michel Habib, and Christophe Paul. Revisiting T. Uno and M. Yagiura's algorithm. In *Lecture notes in computer science*, volume 3827, pages 146–155. ISAAC, Springer, 2005.

[21] Josef Cibulka. On constants in the Füredi–Hajnal and the Stanley–Wilf conjecture. *Journal of Combinatorial Theory, Series A*, 116(2):290–302, 2009.

[22] Sergi Elizalde. *Statistics on pattern-avoiding permutations*. PhD thesis, MIT, 2004.

[23] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[24] Steffen Heber and Jens Stoye. Finding all common intervals of k permutations. In *12th Annual Symposium Combinatorial Pattern Matching, (CPM 2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 207–218. Springer Verlag, 2001.

[25] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[26] Sergey Kitaev and Toufik Mansour. A survey on certain pattern problems. Preprint available at `https://personal.cis.strath.ac.uk/sergey.kitaev/publications.html`, 2003.

[27] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading MA, 3rd edition, 1973.

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

[29] Adeline Pierrot and Dominique Rossin. Simple permutation poset. Preprint available at `http://arxiv.org/abs/1201.3119`, 2012.

[30] Vincent Vatter. Enumeration schemes for restricted permutations. *Combinatorics, Probability and Computing*, 17(1):137–159, 2008.

[31] Vincent Vatter. Permutation classes of every growth rate above 2.48188. *Mathematika*, 56:182–192, 2010.

[32] Vincent Vatter. Small permutation classes. *Proceedings of the London Mathematical Society*, 103:879–921, 2011.

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