An algorithm computing combinatorial specifications of permutation classes

Frédérique Bassino\textsuperscript{a,*}, Mathilde Bouvel\textsuperscript{b}, Adeline Pierrot\textsuperscript{c}, Carine Pivoteau\textsuperscript{d}, Dominique Rossin\textsuperscript{e}

\textsuperscript{a}Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, F-93430 Villetaneuse, France  
\textsuperscript{b}Institut für Mathematik, Universität Zürich, Winterthurerstr. 190, CH-8057 Zürich, Switzerland 
\textsuperscript{c}LRI, Université Paris-Sud, Bat. 650 Ada Lovelace, 91405 Orsay Cedex, France 
\textsuperscript{d}LIGM UMR 8049, Université Paris Est and CNRS, Marne-la-Vallée, France  
\textsuperscript{e}LIX UMR 7161, École Polytechnique and CNRS, Palaiseau, France

Abstract
This article presents a methodology that automatically derives a combinatorial specification for a permutation class \( C \), given its basis \( B \) of excluded patterns and the set of simple permutations in \( C \) when these sets are both finite. This is achieved considering both pattern avoidance and pattern containment constraints in permutations. The obtained specification yields a system of equations satisfied by the generating function of \( C \), this system being always positive and algebraic. It also yields a uniform random sampler of permutations in \( C \). The method presented is fully algorithmic.

Keywords: Pattern avoidance, Permutation class, Combinatorial specification, Generating function, Random sampler

1. Introduction
Permutation classes (and the underlying pattern order on permutations) were defined in the seventies, and since then the enumeration of specific permutation classes (i.e., sets of permutations closed under taking patterns) has received a lot of attention. In this context, as in many in combinatorics, a recursive description of the permutations belonging to the class is often the key towards their enumeration. This recursive description is \textit{a priori} specific to the class studied. But more recently, the substitution decomposition (along with other general frameworks, see [30, and references therein]) has been introduced for the study of permutation classes: it provides a general and systematic approach to their study, with a recursive point of view. This tool has already proved useful in solving many enumerative problems [1, 2, 5, 8, among others], but also in other areas like algorithmics [9, 15].

The goal of the current paper is to systematize even more the use of substitution decomposition for describing recursively and enumerating permutation classes. Our main result is an algorithm that computes a combinatorial specification (in the sense of Flajolet

\footnote{Corresponding author}

Preprint submitted to Elsevier February 14, 2017
and Sedgewick [19]) for any permutation class containing finitely many simple permutations. Note that this problem has been addressed already in [1, 14], however with much less focus on the algorithmic side. Moreover, we introduce in this article a generalization of permutation classes that we call restrictions: while every permutation class is characterized by a set of forbidden patterns, a restriction is described giving a set of forbidden patterns and a set of mandatory patterns. Our algorithm also allows to compute a specification for restrictions containing finitely many simple permutations.

The article is organized as follows. We start by recalling the necessary background in Section 2: permutation classes, substitution decomposition, and the symbolic method. Section 3 gives a more detailed presentation of our results. Here, we dedicate specific attention to explaining the differences between our work and those of [1, 14], and to putting our result in a more global algorithmic context (namely, we describe an algorithmic chain from the basis $B$ of a class $C$ to random sampling of permutations in $C$).

With the next sections, we enter the technical part of our work. After briefly solving the case of substitution-closed classes in Section 4, we explain in two steps how to obtain a combinatorial specification for other classes $C$. Section 5 gives an algorithm producing an ambiguous system of combinatorial equations describing $C$. Next, Section 6 describes how to adapt this algorithm to obtain a combinatorial specification for $C$. Finally, Section 7 illustrates the whole process on examples.

2. Some background on permutations and combinatorial specifications

2.1. Permutation patterns and permutation classes

A permutation $\sigma$ of size $|\sigma| = n$ is a bijective map from $[1..n] = \{1, \ldots, n\}$ to itself. We represent a permutation by a word $\sigma = \sigma_1\sigma_2\ldots\sigma_n$, where each letter $\sigma_i$ denotes the image of $i$ under $\sigma$. We denote $\varepsilon$ the only permutation of size 0; $\varepsilon$ is also called the empty permutation.

**Definition 2.1.** For any sequence $s$ of $k$ distinct integers, the normalization of $s$ is the permutation $\pi$ of size $k$ which is order-isomorphic to $s$, i.e., $s_\ell < s_m$ whenever $\pi_\ell < \pi_m$.

For any permutation $\sigma$ of size $n$, and any subset $I = \{i_1, \ldots, i_k\}$ of $\{1, \ldots, n\}$ with $i_1 < \ldots < i_k$, $\sigma_I$ denotes the permutation of size $k$ obtained by normalization of the sequence $\sigma_{i_1}\ldots\sigma_{i_k}$.

**Definition 2.2.** A permutation $\pi$ is a pattern of a permutation $\sigma$ if and only if there exists a subset $I$ of $\{1, \ldots, |\sigma|\}$ such that $\sigma_I = \pi$. We also say that $\sigma$ contains or involves $\pi$, and we write $\pi \preceq \sigma$. A permutation $\sigma$ that does not contain $\pi$ as a pattern is said to avoid $\pi$.

**Example 2.3.** The permutation $\sigma = 316452$ contains the pattern $2431$ whose occurrences are $3642$ and $3652$. But $\sigma$ avoids the pattern $2413$ as none of its subsequences of length 4 is order-isomorphic to $2413$.

The pattern containment relation $\preceq$ is a partial order on permutations, and permutation classes are downsets under this order. In other words:

**Definition 2.4.** A set $C$ of permutations is a permutation class if and only if for any $\sigma \in C$, if $\pi \preceq \sigma$, then we also have $\pi \in C$. 

2
Throughout this article, we take the convention that a permutation class only contains permutations of size \( n \geq 1 \), i.e., \( \varepsilon \notin \mathcal{C} \) for any permutation class \( \mathcal{C} \).

Every permutation class \( \mathcal{C} \) can be characterized by a unique antichain \( B \) (i.e., a unique set of pairwise incomparable elements) such that a permutation \( \sigma \) belongs to \( \mathcal{C} \) if and only if it avoids every pattern in \( B \) (see for example [1]). The antichain \( B \) is called the basis of \( \mathcal{C} \), and we write \( \mathcal{C} = Av(B) \). The basis of a class \( \mathcal{C} \) may be finite or infinite; it is described as the permutations that do not belong to \( \mathcal{C} \) and that are minimal in the sense of \( \preceq \) for this criterion.

### 2.2. Simple permutations and substitution decomposition of permutations

The description of permutations in the framework of constructible structures (see Section 2.3) that will be used in this article relies on the substitution decomposition of permutations. Substitution decomposition is a general method, adapted to various families of discrete objects [24], that is based on core items and relations, and in which every object can be recursively decomposed into core objects using relations. In the case of permutations, the core elements are simple permutations and the relations are substitutions.

**Definition 2.5.** An interval of a permutation \( \sigma \) of size \( n \) is a non-empty subset \( \{i, \ldots, (i + \ell - 1)\} \) of consecutive integers of \( \{1, \ldots, n\} \) whose images by \( \sigma \) also form a set of consecutive integers. The trivial intervals of \( \sigma \) are \( \{1\} \), \( \{n\} \) and \( \{1, \ldots, n\} \). The other intervals of \( \sigma \) are called proper.

**Definition 2.6.** A block (resp. normalized block) of a permutation \( \sigma \) is any sequence \( \sigma_{i_1} \ldots \sigma_{i_m} \) (resp. any permutation \( \sigma_I \)) for \( I = \{i_1, \ldots, i_m\} \) an interval of \( \sigma \).

**Definition 2.7.** A permutation \( \sigma \) is simple when it is of size at least 4 and it contains no interval, except the trivial ones.

Note that no permutation of size 3 has only trivial intervals (so that the condition on the size is equivalent to “at least 3”).

**Remark 2.8.** The permutations 1, 12 and 21 also have only trivial intervals, and are considered simple in many articles. Nevertheless, for our notational convenience in this work, we prefer to consider that they are not simple.

For a detailed study of simple permutations, in particular from an enumerative point of view, we refer the reader to [1, 4, 16]. Let us only mention that the number of simple permutations of size \( n \) is asymptotically equivalent to \( \frac{n!}{e^2} \) as \( n \) grows.

Let \( \sigma \) be a permutation of size \( n \) and \( \pi^1, \ldots, \pi^n \) be \( n \) permutations of size \( p_1, \ldots, p_n \) respectively. Define the substitution \( \sigma[\pi^1, \pi^2, \ldots, \pi^n] \) of \( \pi^1, \pi^2, \ldots, \pi^n \) in \( \sigma \) (also called inflation in [1]) to be the permutation obtained by concatenation of \( n \) sequences of integers \( S^1, \ldots, S^n \) from left to right, such that for every \( i, j \), the integers of \( S^i \) form a block, are ordered in a sequence order-isomorphic to \( \pi^i \), and \( S^i \) consists of integers smaller than \( S^j \) if and only if \( \sigma_i < \sigma_j \). The interested reader may find a formal definition in [26, Definition 0.25]. When a permutation \( \tau \) may be written as \( \tau = \sigma[\pi^1, \pi^2, \ldots, \pi^n] \), we also say that \( \sigma[\pi^1, \pi^2, \ldots, \pi^n] \) provides a block decomposition of \( \tau \).

When substituting in $\sigma = 12$ or $21$, we often use $\oplus$ (resp. $\ominus$) to denote the permutation $12$ (resp. $21$).

Definition 2.10. A permutation $\pi$ is $\oplus$-indecomposable (resp. $\ominus$-indecomposable) if it cannot be written as $\oplus[\pi^1, \pi^2]$ (resp. $\ominus[\pi^1, \pi^2]$).

Simple permutations, together with $\oplus$ and $\ominus$, are enough to describe all permutations through their substitution decomposition:

Theorem 2.11 (Proposition 2 of [1]). Every permutation $\pi$ of size $n$ with $n \geq 2$ can be uniquely decomposed as either:

- $\oplus[\pi^1, \pi^2]$, with $\pi^1$ $\oplus$-indecomposable,
- $\ominus[\pi^1, \pi^2]$, with $\pi^1$ $\ominus$-indecomposable,
- $\sigma[\pi^1, \pi^2, \ldots, \pi^k]$ with $\sigma$ a simple permutation of size $k$.

Remark 2.12. The simple permutation $\sigma$ in the third item of Theorem 2.11 is a pattern of the permutation $\pi$. Hence, as soon as $\pi$ belongs to some permutation class $C$, then so does $\sigma$.

Theorem 2.11 provides the first step in the decomposition of a permutation $\pi$. To obtain its full decomposition, we can recursively decompose the permutations $\pi^i$ in the same fashion, until we reach permutations of size 1. This recursive decomposition can naturally be represented by a tree, that is called the substitution decomposition tree (or decomposition tree for short) of $\pi$.

Definition 2.13. The substitution decomposition tree $T$ of a permutation $\pi$ is the unique ordered tree encoding the substitution decomposition of $\pi$, where each internal node is either labeled by $\oplus$, $\ominus$ – those nodes are called linear – or by a simple permutation $\sigma$ – prime nodes.

Note that in decomposition trees, linear nodes are always binary, and the left child of a node labeled by $\oplus$ (resp. $\ominus$) may not be labeled $\oplus$ (resp. $\ominus$), since $\pi^1$ is $\oplus$-indecomposable (resp. $\ominus$-indecomposable) in the first (resp. second) item of Theorem 2.11.

Example 2.14. The permutation $\pi = 6987311541017121416131512$ can be recursively decomposed as

\[
\pi = 2413[476519328,1,12,35241] = 2413[31524[\oplus[1, \ominus[1, \oplus[1, 1], 1]], 1, 1, \ominus[1, 1], 1]] , 1, \oplus[1, 1], \ominus[2413[1, 1, 1, 1], 1] \]

and its decomposition tree is given in Figure 1.

Definition 2.15. The substitution closure $\hat{C}$ of a permutation class $C$ is defined as $\cup_{k \geq 1} C^k$ where $C^1 = C$ and $C^{k+1} = \{\sigma[\pi^1, \ldots, \pi^n] \mid \sigma \in C$ and $\pi^i \in C^k$ for any $i$ from 1 to $n = |\sigma|\}$. 
Because simple permutations contain no proper intervals, we have:

**Remark 2.16.** For any class $\mathcal{C}$, the simple permutations in $\hat{\mathcal{C}}$ are exactly the simple permutations in $\mathcal{C}$.

Consequently, for any permutation class $\mathcal{C}$, this allows to describe $\hat{\mathcal{C}}$ as the class of all permutations whose decomposition trees can be built on the set of nodes $\{\oplus, \ominus\} \cup S_\mathcal{C}$, where $S_\mathcal{C}$ denotes the set of simple permutations in $\mathcal{C}$ (if 12 and 21 belong to $\mathcal{C}$; otherwise we have to remove $\oplus$ or $\ominus$ from the set of nodes).

**Definition 2.17.** A permutation class $\mathcal{C}$ is substitution-closed if $\mathcal{C} = \hat{\mathcal{C}}$, or equivalently if for every permutation $\sigma$ of $\mathcal{C}$, and every permutations $\pi_1, \pi_2, \ldots, \pi_n$ of $\mathcal{C}$ (with $n = |\sigma|$), the permutation $\sigma[\pi_1, \pi_2, \ldots, \pi_n]$ also belongs to $\mathcal{C}$.

Like before, a substitution-closed permutation class can therefore be seen as the set of decomposition trees built on the set of nodes $\{\oplus, \ominus\} \cup S_\mathcal{C}$ (if 12 and 21 belong to $\mathcal{C}$; but otherwise $\mathcal{C}$ is trivial and has at most one permutation of each size).

**Remark 2.18.** In [1], it is proven that substitution-closed permutation classes can be characterized as the permutation classes $Av(B)$ whose basis $B$ contains only simple permutations (or maybe 12 or 21 for trivial classes).

### 2.3. From combinatorial specifications to generating functions and random samplers

Let us leave aside permutations for now, and review some basics of the symbolic method about constructible structures and their description by combinatorial specifications. We will see in Theorem 6.4 (p.28) that the classes of permutations we are interested in fit in this general framework.

A class $\mathcal{C}$ of combinatorial structures is a set of discrete objects equipped with a notion of size: the size is a function of $\mathcal{C} \to \mathbb{N}$ denoted $|\cdot|$ such that for any $n$ the number of objects of size $n$ in $\mathcal{C}$ is finite.

Among the combinatorial structures, we focus on constructible ones, from the framework introduced in [19]. Basically, a constructible combinatorial class is a set of structures that can be defined from atomic structures of size 1 (denoted by $\mathcal{Z}$), possibly structures...
of size 0 (denoted by $E$), and assembled by means of admissible constructors. While a complete list of these combinatorial constructors is given in [19], we only use a (small) subset of them: the disjoint union, denoted by $\uplus$ or $+$ (we may also use the notation $\sum$), to choose between structures; and the Cartesian product, denoted by $\times$, to form pairs of structures. More formally, a constructible combinatorial class is one that admits a combinatorial specification.

**Definition 2.19.** A combinatorial specification for a combinatorial class $C_1$ is an equation or a system of equations of the form

\[
\begin{align*}
C_1 &= H_1(E, Z, C_1, C_2, \ldots, C_m), \\
C_2 &= H_2(E, Z, C_1, C_2, \ldots, C_m), \\
&\vdots \\
C_m &= H_m(E, Z, C_1, C_2, \ldots, C_m),
\end{align*}
\]

where each $H_i$ denotes a term built from $C_1, \ldots, C_m, Z$ and $E$ using admissible constructors.

For example, the equation $I = E + Z \times I$ describes a class $I$ whose elements are finite sequences of atoms.

In this framework, the size of a combinatorial structure is its number of atoms ($Z$) and from there, combinatorial structures can be counted according to their size. The size information for a whole combinatorial class, say $C$, is encoded by its ordinary generating function\(^1\), which is the formal power series $C(z) = \sum_{n \geq 0} c_n z^n$ where the coefficient $c_n$ is the number of structures of size $n$ in $C$. Note that we also have $C(z) = \sum_{\pi \in C} z^{\mid \pi \mid}$.

Combinatorial specifications of combinatorial classes may be automatically translated into systems defining their generating function (possibly implicitly). This system is obtained by means of a dictionary that associates an operator on generating functions to each admissible constructor. The complete dictionary is given in [19], together with the proof that this translation from constructors of combinatorial classes to operators on their generating functions is correct. Here, we only use the constructors disjoint union and Cartesian product, which are respectively translated to sum and product of generating functions.

A lot of information can be extracted from such functional systems; in particular, one can compute as many coefficients of the series as required, and [19] provides many tools to get asymptotic equivalents for these coefficients.

Combinatorial specifications may also be automatically translated into uniform random samplers of objects in the class described by the specification. Indeed, a specification can be seen as a (recursive) procedure to produce combinatorial objects, and randomizing the choices made during this procedure transforms the specification into a random sampler. To ensure that such random samplers are uniform (i.e. that for any $n$, two

---

\(^1\)We do not use exponential but ordinary generating functions to count pattern-avoiding permutations. First, note that the corresponding exponential generating functions would have infinite radii of convergence, pattern-avoiding permutations of size $n$ being always less than $c^n$ for some constant $c$ [25]. The use of ordinary generating functions is moreover very natural since our work is based on an encoding of permutations by trees built on a finite set of nodes.
objects of the same size \( n \) have the same probability of being produced), two methods have been developed: the recursive method [20] and the Boltzmann method [18]. In the first one, the coefficients of the generating functions are used for the probabilistic choices to ensure uniformity, making this method well-adapted for generating a large sample of objects of relatively small size: this requires to compute only once a relatively small number of coefficients. The focus of the second one is to achieve efficiently the generation of very large objects, with a small tolerance (of a few percents) allowed on their size. Coefficients of the generating functions are not needed in Boltzmann samplers, but rather the generating functions themselves. More precisely, the value of the generating function at a given point needs to be computed, and this is solved in [28].

3. Our results in existing context

3.1. Our contributions, and comparison with [1, 14]

The goal of the present work is to solve algorithmically a combinatorial problem on permutation classes: computing in an automatic way a combinatorial specification for any given class, under some conditions specified below. We first have to determine how to describe the permutation class \( \mathcal{C} \) in input. In all what follows, we will suppose that \( \mathcal{C} \) is given by its basis \( B \) of excluded patterns and the set \( \mathcal{S}_\mathcal{C} \) of simple permutations in \( \mathcal{C} \), assuming that both these sets are finite. Note that from [1, Theorem 9] the basis of \( \mathcal{C} \) is necessarily finite when \( \mathcal{C} \) contains finitely many simple permutations. On the other hand, from [17, 13], it is enough to know \( B \) to decide whether \( \mathcal{S}_\mathcal{C} \) is finite and (in the affirmative) to compute \( \mathcal{S}_\mathcal{C} \) [27].

Our work is a continuation of the main result (Theorem 10) of [1]: every permutation class containing finitely many simples has an algebraic generating function. The main step in the proof of this result is to construct a system of combinatorial equations describing \( \mathcal{C} \) using the substitution decomposition, and more precisely by propagation of pattern avoidance constraints in the decomposition trees. Although the proof is in essence constructive, there is still some work to be done to fully automate this process. In Section 5, we review their method, going deeper in the details of the construction. This allows us to bring their methodology to a full algorithm – see Algorithm \textsc{AmbiguousSystem} (p.16) and Theorem 5.1 (p.16).

It is important to note that the description of \( \mathcal{C} \) obtained in this way is not a combinatorial specification, since it is \textit{a priori} ambiguous (that is to say, unions are not necessarily disjoint). Nevertheless, as it is done in [1], this ambiguous system can be used for proving the algebraicity of the generating function of \( \mathcal{C} \), and even allows its computation (or implicit determination, in less favorable cases): it is enough to apply the inclusion-exclusion principle. The advantage of a specification over an ambiguous system will be discussed in Section 3.2 (fifth step, p.12).

The algebraicity result of [1] was re-proved in [14] and extended to generating functions of some subsets of classes \( \mathcal{C} \) with finitely many simples: the alternating permutations in \( \mathcal{C} \), the even ones, the involutions in \( \mathcal{C} \), ... For our purpose, those extensions are less important than the alternative proof of the main result of [1]: indeed, this second proof describes a method to build a combinatorial specification for \( \mathcal{C} \). Essential to this proof are \textit{query-complete sets}, whose definition we recall.
Definition 3.1. A property is any set $P$ of permutations. A permutation $\pi$ is said to satisfy $P$ when $\pi \in P$. A set $P$ of properties is query-complete if, for every simple permutation $\sigma$ (and also for $\sigma = \oplus$ or $\ominus$) and for every property $P \in P$, it can be decided whether $\sigma[\alpha_1, \ldots, \alpha_m]$ satisfies $P$ knowing only which properties of $P$ are satisfied by each $\alpha_i$.

The proof of the main result (Theorem 1.1) of [14] shows that a combinatorial specification for a permutation class $C$ with finitely many simples can be obtained from any finite query-complete set $P$ such that $C \in P$. More precisely, this combinatorial specification consists of three types of equations, described below (this is reduced to two types by plugging the third one into the second one). Recall that $S_C$ denotes the set of simple permutations in $C$. Note that all unions below are finite, since $P$ and $S_C$ are finite by assumption.

- First, $C$ is written as the disjoint union
  $$C = \biguplus C_X,$$
  where the union runs over all subsets $X$ of $P$ containing $C$ with $C_X$ denoting the set of permutations that satisfy every property in $X$ and do not satisfy any property in $P \setminus X$.

- Second, for any such set $X$, the substitution decomposition allows to write
  $$C_X = 1_X \uplus C_X^{\oplus} \uplus C_X^{\ominus} \uplus \biguplus_{\sigma \in S_C} C_{X}^{\sigma},$$
  where $1_X$ is either the set $\{1\}$ if the permutation 1 belongs to $C_X$, or the empty set otherwise, and where $C_X^{\sigma}$ is the subset of $C_X$ of permutations whose decomposition tree has root $\sigma$.

- And third, for $\sigma \in \{\oplus, \ominus\} \cup S_C$ and $X$ as above, the fact that $P$ is query-complete allows to express $C_X^{\sigma}$ as
  $$C_X^{\sigma} = \biguplus \sigma[C_X, \ldots, C_X]$$
  where the union is over the set $E_{X, \sigma}$ of all $m$-tuples $(X_1, \ldots, X_m)$ of subsets of $P$ such that if, for every $i \in [1..m]$, it holds that $\alpha_i \in C_{X_i}$ then $\sigma[\alpha_1, \ldots, \alpha_m] \in C_{X}$. In the case where $\sigma = \oplus$ (resp. $\ominus$), to ensure uniqueness of the decomposition, we further need to enforce that $X_i$ contains the property of being $\oplus$-indecomposable (resp. $\ominus$-indecomposable). Without loss of generality, we can assume that these properties are in $P$.

Note that the number of equations in the specification obtained depends exponentially on the size of $P$, since there is at least one equation for each subset of $P$ containing $C$. Similarly, the number of terms of the union defining some $C_{X}^{\sigma}$ may be exponential in the size of $P$, since the union is over $m$-tuples of subsets of $P$ (with $m = |\sigma|$).

We point out that the above specification is not fully explicit (even assuming that $P$ is given), since there is no explicit description of the sets $E_{X, \sigma}$. As explained in the proof of Lemma 2.1 of [14], the sets $E_{X, \sigma}$ may be described using so-called lenient inflations,
which are intimately linked with the *embeddings by blocks* of [1]. But neither [14] nor [1] discuss their effective computation. We will return to this problem later in this section.

For any permutation class $C = \text{Av}(B)$ with finitely many simples, the authors of [14] provide a finite query-complete set that contains $C$, and conclude that there is a combinatorial specification for any such $C$. More precisely, the class $C$ being described as $\cap_{\beta \in B} \text{Av}(\beta)$, they rather consider separately every principal class $\text{Av}(\beta)$ for all $\beta \in B$, and define a finite query-complete set $P_\beta$ containing it. This is essentially their Lemma 2.1. The query complete set associated with $C$, denoted $P_C$, is then obtained taking the union of all $P_\beta$. It consists of the following properties: the set of $\oplus$-indecomposable permutations, the set of $\ominus$-indecomposable permutations, and the set $\text{Av}(\rho)$ for every permutation $\rho$ which is a pattern of some $\beta \in B$. Thus $P_C$ is often a big set.

It should be noticed that the query-complete sets that are used in the examples of [14, Section 4] are however strictly included in the set $P_C$. These smaller query-complete sets are better, since they result in specifications with fewer equations and unions having fewer terms than the ones that would be obtained applying to the letter the specializations of the proofs. But there are no indications in [14] on how these smaller query-complete sets were computed, nor on how this could generalize to other examples. It should be noticed that in the examples of [14], the class is either substitution-closed, or contains no simple permutations.

To summarize, the proof of the main result of [14] gives a general method to compute a specification for a permutation class having finitely many simple permutations. But there is still some work to be done to fully automate this process, and the specification obtained would be very big in general. Moreover, an algorithm using this method would have a lot of computations to do (the computation of all the sets $E_{X,\sigma}$). On the other hand, the *ad hoc* constructions of the examples of [14] show that using the specificity of a particular permutation class, it is possible to obtain shorter specifications with fewer computations.

Our main contribution is to give an algorithm to compute a specification for any permutation class having finitely many simples, using a different approach – see Algorithm Specification (p.26) and Theorem 6.4 (p.28). Our method is general (unlike the *ad hoc* methods used in the examples of [14]) but nevertheless uses the specificity of the permutation class given in input (unlike the method described in the proofs of [14]) in order to do less computations and to have fewer equations in the specification.

Even if the method we use and the specification we produce are not exactly the ones presented by [14] and reviewed above, they have some similarities. When considering the partition of $C$ into $\cup C_X$ shown above, [14] is looking at a very fine level of details, where a coarser level could be enough. With our method, we consider the partition of $C$ which is the coarsest possible to allow the derivation of a specification. In practice, if two sets $X$ and $X'$ are such that $C_X$ and $C_{X'}$ appear as $C_X \cup C_{X'}$ everywhere in the specification resulting from [14], our specification will have only one term instead of these two, representing $C_X \cup C_{X'}$. Of course, this holds for unions with more terms as well. Considering fewer sets $X$ results in fewer equations in the specification, fewer terms in the unions and fewer sets $E_{X,\sigma}$. Since the computation of the $E_{X,\sigma}$ amounts to computing the specification from the query-complete set, the algorithmic complexity for computing the specification with our method is hereby reduced, compared to what a formalized algorithm of the approach of [14] would give.
Concretely, our algorithm computes a query-complete set, the sets $E_{X,\sigma}$ associated, and the specification in parallel, whereas the method of [14] is to first compute a query-complete set and then deduce a specification. Note that in our presentation, the result of our algorithm is only the specification, but it contains also implicitly the description of the query-complete set and of the $E_{X,\sigma}$.

To obtain the announced coarsest partition of $C$, and the subsequent specification with as few equations as possible that it yields, we proceed as follows. We use the same guideline as in [1] for computing a possibly ambiguous combinatorial system describing $C$ (however making this approach effective): the essential idea is to use the substitution decomposition and to propagate pattern avoidance constraints in the decomposition trees. We get rid of the ambiguity by introducing complement sets, but only when they are needed (the method in the proofs of [14] can be seen as somehow introducing all complement sets at once). In practice, it means that we are not only propagating avoidance constraints in decomposition trees, but also containment constraints. It will be clear in Sections 5 and 6 that even though the purposes of those two types of constraint are opposite, the ways to propagate them are very similar, an essential step being the effective computation of the embeddings/lenient inflations mentioned in [1, 14]. The method used to explicitly determine which avoidance/containment constraints are necessary and to effectively propagate them in the trees is completely new with respect to [1, 14].

To conclude on our contributions compared with those of [1, 14], our work describes how to obtain a combinatorial specification for any class having finitely many simple permutations. Contrary to [1, 14], our work is fully algorithmic. Moreover, we develop a method allowing to have fewer equations in the specification and to have a better efficiency compared to what a formalized algorithm of the approach of [14] would give.

### 3.2. An algorithmic chain from $B$ to random permutations in $Av(B)$

Our main result (that is, the algorithmic computation of specifications for permutation classes with finitely many simples) can and should be viewed in the context of other recent algorithms from the literature. Together, they provide a full algorithmic chain starting with the finite basis $B$ of a permutation class $C$, and computing a specification for $C$, from which it is possible to sample permutations in $C$ uniformly at random. Figure 2 shows an overview of this algorithmic chain, and we present its main steps below. Note that this procedure may fail to compute its final result, namely when $C$ contains an infinite number of simple permutations, this condition being tested algorithmically.

We have chosen that the permutation class in input of our procedure should be given by its basis $B$, that we require to be finite. This does not cover the whole range of permutation classes, but it is one of the possible ways to give a finite input to our algorithm. There are of course other finite descriptions of permutation classes, even of some with infinite basis (by a recognition procedure for example). The assumption of the description by a finite basis has been preferred for two reasons: first, it encompasses most of the permutation classes that have been studied; and second, it is a necessary condition for classes to contain finitely many simple permutations (see [1, Theorem 9]) and hence for our algorithm to succeed.

**First step: Finite number of simple permutations.**
First, we check whether $C = Av(B)$ contains only a finite number of simple permutations. This is achieved using algorithms of [12] when the class is substitution-closed and of [13]
Is there a finite number of simple permutations in the class $C = \text{Av}(B)$?

$O(n \log n)$

**B**: finite basis of excluded patterns

- Contains only simple permutations
- $\text{Av}(B)$ is substitution-closed

**NO**

**YES**

Figure 2: The full algorithmic chain starting from the basis $B$ of a permutation class $C$, with complexities given with respect to $n = \sum_{\beta \in B} |\beta|$, $k = |B|$, $p = \max\{|\beta| : \beta \in B\}$, $N = |S_C|$ and $\ell = \max\{|\pi| : \pi \in S_C\}$ where $S_C$ is the set of simple permutations of $C$.

otherwise. The complexity of these algorithms is respectively $O(n \log n)$ and $O(n \log n + p^{2k})$, where $n = \sum_{\beta \in B} |\beta|$, $p = \max\{|\beta| : \beta \in B\}$ and $k = |B|$.

**Second step: Computing simple permutations.**

The second step of the algorithm is the computation of the set of simple permutations $S_C$ contained in $C = \text{Av}(B)$, when we know it is finite. Again, when $C$ is substitution-closed, $S_C$ can be computed by an algorithm that is more efficient than in the general case. The two algorithms are described in [27], and their complexity depends on the output:

$O(N \cdot \ell^4)$ in general and $O(N \cdot \ell^4)$ for substitution-closed classes, with $N = |S_C|$, $p = \max\{|\beta| : \beta \in B\}$, $\ell = \max\{|\pi| : \pi \in S_C\}$ and $k = |B|$.

In the case of substitution-closed classes, the set of simple permutations in $C$ gives an immediate access to a specification for $C$ – see [1] or Theorem 4.3 (p.15). In the general case, finding such a specification is the algorithmic problem that we address in this article.

**Third step: Computing a combinatorial specification.**

This corresponds to the computation of a combinatorial specification for $C$ by propagation of pattern constraints and disambiguation of the equations, as briefly presented in Section 3.1 and described in details in Sections 5 and 6.

From a combinatorial specification for $C$, that we may obtain algorithmically as described above, there are two natural algorithmic continuations (which we have reviewed in Section 2.3):

**Fourth step: Computing the generating function $C(z)$ of $C$.**

With the *dictionary* of [19], a system of equations defining $C(z) = \sum_{n \geq 0} c_n z^n$ is immediately deduced from the specification. Because our specification involves only disjoint unions and Cartesian products, the resulting system is positive and algebraic. In some favorable cases, this system may be solved for $C(z)$ explicitly. Even if it is not the case, many information may still be derived from the system, in particular about the coefficients $c_n$ or the growth rate of the class.
An alternative for the computation of $C(z)$.

As explained in [1] and reviewed earlier in this paper, it is also possible to obtain such a system of equations for $C(z)$ from an ambiguous system describing $C$, applying the inclusion-exclusion principle. In this case, the obtained system is algebraic but with negative terms in general.

**Fifth step: Random sampling of permutations in $C$.**

In Section 2.3, we have reviewed the principles that allow, in the same fashion as the dictionary of [19], to translate the combinatorial specification for $C$ into uniform random samplers of permutations in $C$. Remark that this translation is possible only with a specification, i.e. a positive unambiguous system describing $C$. Indeed, whereas adapted when considering generating functions (where subtraction is easily handled), the inclusion-exclusion principle cannot be applied for random generation (since “subtracting combinatorial objects” is not an option in a procedure to produce them).

To illustrate that this algorithmic chain is effective, we present in Section 7 how our algorithms run on examples. We also show some observations that are produced through it in Figures 3 and 4 below. These figures have been obtained with a prototype implementing our algorithms, that we hope to make available for use by others in the future.

![Figure 3: The shape of permutations in the (not substitution-closed) class $C$ of Section 7.2.](image)

Figure 3 shows the “average diagram” of a permutation in the class $C$ (not substitution-closed) studied in Section 7.2 (p.39). The diagram of a permutation $\sigma$ is the set of points in the plane at coordinates $(i, \sigma_i)$, and the picture in Figure 3 is obtained by drawing uniformly at random 30,000 permutations of size 500 in $C$, and by overlapping their diagrams – the darker a point $(x, y)$ is, the more of these permutations have $\sigma(x) = y$.

Figure 4 shows average diagrams of permutations in the (substitution-closed) class $Av(2413, 3142)$ of separable permutations, and in another substitution-closed class. These diagrams are obtained overlapping the diagrams of 10,000 permutations of size 100 (resp. 500). The representation is however a little different from Figure 3: in these 3D representations, for a point at coordinates $(x, y, z)$, $z$ is the number of permutations such that $\sigma(x) = y$. Leaving aside the difference in the representation, these figures suggest a very different limit behavior in substitution-closed and not substitution-closed classes.

---

2A Boltzmann sampler for substitution-closed classes is already available here: [http://igm.univ-mlv.fr/~pivoteau/Permutations/](http://igm.univ-mlv.fr/~pivoteau/Permutations/). The implementation is in Maple, an example of use is given in the worksheet.
Looking at these diagrams, a natural question is then to describe the average shape of permutations in classes. This is a question which has received quite a lot of attention lately, especially for classes $Av(\tau)$ for $\tau$ of size 3, see [7, 21, 22, 23]. Inspired by Figure 4, some of us (in collaboration with V. Féray and L. Gerin) have described the limit shape of separable permutations in [10], thus explaining the first diagram of Figure 4. As we discuss in [10], we are working on generalizing this result to substitution-closed classes, which would also explain the second diagram of Figure 4.

3.3. Perspectives

As described in Section 3.2, our main result combines with previous works to yield an algorithm that produces, for any class $Av(B)$ containing finitely many simple permutations, a recursive (resp. Boltzmann) uniform random sampler. When generating permutations with such samplers, complexity is measured with respect to the size of the permutation produced and is quasilinear (resp. quadratic but can be made linear using classical tricks and allowing a small variation on the size of the output permutation [18]). However, the complexity is not at all measured with respect to the number of equations in the specification nor with respect to the number of terms in each equation. In our context, where the specifications are produced automatically, and potentially contain a large number of equations/terms, this dependency is of course relevant, and opens a new direction in the study of random samplers.

In addition to providing inspiration for the study of random permutations, our algorithmic chain has other applications. Indeed, the specifications obtained could also be used to compute or estimate growth rates of permutation classes. Moreover, the computed specifications could possibly be used to provide more efficient algorithms to test membership of a permutation to a class.

We should also mention that our procedure fails to be completely general. Although the method is generic and algorithmic, the classes that are fully handled by the algorithmic process are those containing a finite number of simple permutations. From [1], such
classes are finitely based. And since there are countably many such permutation classes, only a very small subset of the (uncountably many) permutation classes is fully covered by our method. However note that even if a class \( C \) contains an infinite number of simple permutations, we can use our approach to perform random generation of permutations in \( C \). Indeed to generate permutations of size at most \( n \) in \( C \), it is enough to consider simple permutations of size up to \( n \). This approach may be relevant mostly when the class has a small number of simple permutations of each size.

To enlarge the framework of application of our algorithm computing specifications, we could explore the possibility of extending it to permutation classes that contain an infinite number of simple permutations, but that are finitely described. A family of such classes is considered in [6], where the finite basis and algebraicity results of [1] are extended from classes with finitely many simple permutations to subclasses of substitution closures of geometrically griddable classes (we refer the reader to [6] for definitions). The proofs in [6] involve similar techniques as in [14] (including query-complete sets), but also other techniques. In particular, they heavily rely on the results of [3], which are non-constructive. Making all the process in [3] and [6] constructive, and then turning it into an effective procedure, it may be possible to extend our algorithms to all classes considered in [6]. But this is far from straightforward, and beyond the scope of the present work. Note that, with such an improvement, more classes would enter our framework, but it would be hard to leave the algebraic case.

4. Combinatorial specification of substitution-closed classes

In this section, we recall how to obtain a combinatorial specification for substitution-closed classes having finitely many simple permutations.

Recall that we denote by \( \hat{C} \) the substitution closure of the permutation class \( C \), and that \( C \) is substitution-closed when \( C = \hat{C} \), or equivalently when the permutations in \( C \) are all the permutations whose decomposition trees have internal nodes labeled by \( \oplus, \ominus \) or any simple permutation of \( C \).

For the purpose of this article, we additionally introduce the following notation:

**Definition 4.1.** For any set \( A \) of permutations, \( A^+ \) (resp. \( A^- \)) denotes the set of permutations of \( A \) that are \( \oplus \)-indecomposable (resp. \( \ominus \)-indecomposable) and \( S_A \) denotes the set of simple permutations of \( A \).

Theorem 2.11 (p.4) directly yields the following proposition:

**Proposition 4.2** (Lemma 11 of [1]). Let \( C = \hat{C} \) be a substitution-closed class\(^3\). Then \( \hat{C} \) satisfies the following system of equations, denoted \( \mathcal{E}_{\hat{C}} \):

\[
\begin{align*}
\hat{C} &= 1 \cup \ominus[\hat{C}^+, \hat{C}] \cup \ominus[\hat{C}^-, \hat{C}] \cup \bigcup_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}] \\
\hat{C}^+ &= 1 \cup \ominus[\hat{C}^-, \hat{C}] \cup \bigcup_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}] \\
\hat{C}^- &= 1 \cup \ominus[\hat{C}^+, \hat{C}] \cup \bigcup_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}].
\end{align*}
\]

---

\(^3\)that contains 12 and 21; it will be the case until the end of the article and will not be recalled again.
Note that by Remark 2.16, \( S_C = S_\hat{C} \). By uniqueness of the substitution decomposition, unions are disjoint and so Equations (1) to (3) describe unambiguously the substitution-closed class \( \hat{C} \). Hence, Proposition 4.2 can be transposed in the framework of constructible structures as follows:

**Theorem 4.3.** Let \( C \) be a substitution-closed class. Then \( C \) can be described as a constructible combinatorial class in the sense of Section 2.3 with the following combinatorial specification, where the \( E_\pi \) for \( \pi \) in \( S_C \) are distinct objects of size 0:

\[
\begin{align*}
C &= Z + E_{\varnothing} \times C^+ \times C + E_{\varnothing} \times C^- \times C + \sum_{\pi \in S_C} E_\pi \times C \times \cdots \times C^{\lfloor |\pi| \rfloor} \\
C^+ &= Z + E_{\varnothing} \times C^- \times C + \sum_{\pi \in S_C} E_\pi \times C \times \cdots \times C^{\lfloor |\pi| \rfloor} \\
C^- &= Z + E_{\varnothing} \times C^+ \times C + \sum_{\pi \in S_C} E_\pi \times C \times \cdots \times C^{\lfloor |\pi| \rfloor}.
\end{align*}
\]

Moreover this system can be translated into an equation for the generating function \( C(z) \):

**Proposition 4.4** (Theorem 12 of [1]). Let \( C \) be a substitution-closed class, with generating function \( C(z) \). Then

\[
C(z)^2 + (S_C(C(z)) - 1 + z)C(z) + S_C(C(z)) + z = 0
\]

with \( S_C(z) \) denoting the generating function that enumerates simple permutations in \( C \), i.e. \( S_C(z) = \sum_{\pi \in S_C} z^{|\pi|} \).

Hence, in the case of a substitution-closed class \( C \) (and for the substitution closure \( \hat{C} \) of any class), the system \( E_\hat{C} \) that recursively describes the permutations in \( \hat{C} \) can be immediately deduced from the set \( S_C \) of simple permutations in \( C \). As soon as \( S_C \) is finite and known, this system is explicit and gives a combinatorial specification.

Our next goal is to describe an algorithm that computes a combinatorial system of equations for a general permutation class \( C \) from the simple permutations in \( C \), like for the case of substitution-closed classes. However, when the class is not substitution-closed, this is not as straightforward as what we have seen in Proposition 4.2, and we provide details on how to solve this general case in the following sections.

5. A possibly ambiguous combinatorial system for permutation classes

In this section, we explain how to derive a system of equations for a class \( C \) with finitely many simple permutations from the combinatorial specification of its substitution closure. Our method follows the guideline of the constructive proof of [1, Theorem 10]. However, unlike [1], we make the whole process fully algorithmic.

The key idea of the method is to describe recursively the permutations in \( C \), replacing the constraint of avoiding the elements of the basis by constraints in the subtrees of the
decomposition tree of permutations in $C$. This is done by computing the embeddings of non-simple permutations $\gamma$ of the basis $B$ of $C$ into simple permutations $\pi$ belonging to the class $C$ (and into $\ominus$ and $\oplus$). These embeddings are block decompositions of the permutations $\gamma$, each (normalized) block being translated into a new avoidance constraint pushed downwards in the decomposition tree. We then need to add new equations in the specification for $C$ to take into account these new constraints.

The main algorithm of this section is $\text{AmbiguousSystem}$ (Algo. 1 below), which uses auxiliary procedures described later on. We prove in Section 5.4 that the result it produces has the properties stated in Theorem 5.1 below, which uses the following notation.

For any permutation class $C$ with basis $B$, we denote by $B\ast$ the subset of non-simple permutations of $B$ and by $S_C$ the set of simple permutations in $C$. Moreover, for any set $A$ of permutations and any set $E$ of patterns, we denote by $A\langle E \rangle$ the set of permutations of $A$ that avoid every pattern in $E$. Note that we have $(A\langle E \rangle)^+ = A^+\langle E \rangle$; the corresponding set is the one of permutations of $A$ that avoid $E$ and that are $\ominus$-indecomposable. The same goes for $A^-\langle E \rangle$.

**Theorem 5.1.** Let $C$ be a permutation class with a finite number of simple permutations whose (finite) basis is $B$. The result of $\text{AmbiguousSystem}(B, S_C)$ is a finite system of combinatorial equations describing $C$. The equations of this system are all of the form $D_0 = 1 \cup \bigcup \pi[D_1, \ldots, D_n]$, where $D_i = C^\delta(B^* \cup B_i)$ with $\delta \in \{+, -, \ominus\}$ and $B_i$ contains only permutations corresponding to normalized blocks of elements of $B^*$. This system contains an equation whose left part is $C$ and is complete, that is: every $D_i$ that appears in the system is the left part of one equation of the system.

**Algorithm 1: $\text{AmbiguousSystem}(B, S_C)$**

**Data:** A finite basis of forbidden patterns defining $C = \text{Av}(B)$ and the finite set $S_C$ of simple permutations in $C$.

**Result:** A system of equations of the form $D_0 = 1 \cup \bigcup \pi[D_1, \ldots, D_n]$ defining $C$.

**begin**

\[ E \leftarrow \text{EqnForClass}(C, B^*) \quad /* \text{See Algo. 5 (p.24)} */ \]

\[ \text{while there is a right-only } C^\delta \langle E \rangle \text{ in some equation of } E \text{ do} \]

\[ E \leftarrow E \cup \text{EqnForClass}(C^\delta, E) \]

**end**

An essential remark is that the obtained combinatorial system may be ambiguous, since it may involve unions of sets that are not always disjoint. We will tackle the problem of computing a non-ambiguous system in Section 6.

It is also to note that the result of $\text{AmbiguousSystem}$ provides a finite query-complete set containing $C$:

---

$^5$Theorem 9 of [1, Theorem 9] ensures that, as soon as $C$ contains finitely many simple permutations, then the basis of $C$ is finite.

$^6$For any set $A$ of permutations, when writing $A^\delta$ for $\delta \in \{+, -, \ominus\}$, we mean that $A^\delta$ is either $A$ or $A^+$ or $A^-$. 

16
Corollary 5.2. Let $E$ be the system of equations output by $\text{ambiguousSystem}(B, S_C)$. Then the set $\{ D : D$ is the left part of some equation of $E \}$ is a query-complete set containing $C$.

Proof. This a direct consequence of the form of $E$ described in Theorem 5.1. □

5.1. A first system of equations

Consider a permutation class $C$, whose basis is $B$ and which is not substitution-closed. We compute a system describing $C$ by adding constraints to the system obtained for $\hat{C}$, as in [1].

Proposition 5.3. Let $C$ be a permutation class, that contains 12 and 21. We have that $C^\delta = \hat{C}^\delta(B^*)$ for $\delta \in \{+, -, \}$. Moreover,

\begin{align*}
\hat{C}(B^*) &= 1 \uplus \bigoplus [\hat{C}^+, \hat{C}^-](B^*) \uplus \bigodot [\hat{C}^-, \hat{C}^+](B^*) \uplus \biguplus_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}](B^*) \\
\hat{C}^+(B^*) &= 1 \uplus \bigoplus [\hat{C}^-, \hat{C}^-](B^*) \uplus \biguplus_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}](B^*) \\
\hat{C}^-(B^*) &= 1 \uplus \bigoplus [\hat{C}^+, \hat{C}^-](B^*) \uplus \biguplus_{\pi \in S_C} \pi[\hat{C}, \ldots, \hat{C}](B^*)
\end{align*}

all these unions being disjoint.

Proof. Let $\sigma \in C$, then $\sigma \in \hat{C}$ and $\sigma$ avoids $B^*$, thus $\sigma \in \hat{C}(B^*)$. Conversely, let $\sigma \in \hat{C}(B^*)$ and let $\pi \in B$. If $\pi \in B^*$ then $\sigma$ avoids $\pi$. Otherwise, $\pi$ is simple and $\pi \notin C$. Because $S_B = S_C$ by Remark 2.16, this implies that $\pi \notin \hat{C}$. Since $\sigma \in \hat{C}$, $\sigma$ avoids $\pi$. Hence $\sigma \in C$.

Finally $\hat{C}(B^*) = C$, thus $\hat{C}^\delta(B^*) = C^\delta$. Then the result follows from Proposition 4.2. □

Equations (4) to (6) do not provide a combinatorial specification because the terms $\pi[\hat{C}, \ldots, \hat{C}]/(B^*)$ are not simply expressed from the terms appearing on the left-hand side of these equations. To solve this problem, instead of decorating all terms on the right-hand side of Equations (1)–(3) with constraint $\langle B^* \rangle$ (like in Equations (4)–(6)), we propagate the constraint $\langle B^* \rangle$ into the subtrees. More precisely, by Lemma 18 of [1], sets $\pi[\hat{C}, \ldots, \hat{C}](B^*)$ can be expressed as union of smaller sets:

$$\pi[\hat{C}, \ldots, \hat{C}](B^*) = \bigcup_{i=1}^k \pi[\hat{C}(E_{i,1}), \hat{C}(E_{i,2}), \ldots, \hat{C}(E_{i,k})]$$

(7)

where the $E_{i,j}$ are sets of permutations which are patterns of some permutations of $B^*$. For instance, with $C = Av(231)$, we have $B^* = \{231\}$, and $\bigcirc[\hat{C}^-, \hat{C}^+](231) = \bigcirc[\hat{C}^-(12), \hat{C}(231)]$. Note that the set $\hat{C}^-(12)$, which is not a part of the initial system, has appeared on the right-hand side of this equation, and we now need a new equation to describe it. In the general case, applying Equation (7) in the system of Proposition 5.3 introduces sets of the form $\hat{C}^\delta(E_{i,j})$ on the right-hand side of an equation of the system that do not appear on the left-hand side of any equation. The reason is that $\{C, C^+, C^\}^-$ is not query-complete in general. Propagating the constraints in the subtrees as described below allows to determine a query-complete set which contains $C$ (see Corollary 5.2).

We call right-only sets the sets of the form $\hat{C}^\delta(E_{i,j})$ which appear only on the right-hand side of an equation of the system. For each such set, we need to add a new equation to the system, starting from Equation (1), (2) or (3) depending on $\delta$, and propagating constraint $E_{i,j}$ instead of $B^*$. This may create new right-only terms in these new equations, and they are treated recursively in the same way. This process
terminates, since the $E_{i,j}$ are sets of patterns of elements of $B^*$, and there is only a finite number of such sets (as $B$ is finite).

The key to the precise description of the sets $E_{i,j}$, and to their effective computation, is given by the embeddings of permutations $\gamma$ that are patterns of some $\beta \in B^*$ into the simple permutations $\pi$ of $C$.

5.2. Embeddings: definition and computation

Recall that $\varepsilon$ denotes the empty permutation, i.e. the permutation of size 0. We take the convention $A(\varepsilon) = \emptyset$.

A generalized substitution (also called lenient inflation in [14]) $\sigma\{\pi^1, \pi^2, \ldots, \pi^n\}$ is defined like a substitution with the particularity that any $\pi^i$ may be the empty permutation. Note that $\sigma[\pi^1, \pi^2, \ldots, \pi^n]$ necessarily contains $\sigma$ whereas $\sigma\{\pi^1, \pi^2, \ldots, \pi^n\}$ may avoid $\sigma$. For instance, the generalized substitution $132\{21, \varepsilon, 1\}$ gives the permutation $213$ which avoids $132$.

Thanks to generalized substitutions, we define the notion of embedding, which expresses how a pattern $\gamma$ can be involved in a permutation whose decomposition tree has a root $\pi$:

**Definition 5.4.** Let $\pi = \pi_1 \ldots \pi_n$ and $\gamma$ be two permutations of size $n$ and $p$ respectively and $P_\gamma$ the set of intervals\(^7\) of $\gamma$, including the trivial ones. An embedding of $\gamma$ in $\pi = \pi_1 \ldots \pi_n$ is a map $\alpha$ from $\{1, \ldots, n\}$ to $P_\gamma$ such that:

- if the intervals $\alpha(i)$ and $\alpha(j)$ are not empty, and $i < j$, then $\alpha(i)$ consists of smaller indices than $\alpha(j)$;
- as a word, $\alpha(1) \ldots \alpha(n)$ is a factorization of the word $1 \ldots |\gamma|$ (which may include empty factors);
- it holds that $\pi\{\gamma_{\alpha(1)}, \ldots, \gamma_{\alpha(n)}\} = \gamma$ (see Definition 2.1 (p.2) for the definition of $\gamma_{\alpha(i)}$).

**Example 5.5.** For any permutations $\gamma$ and $\pi$, $\alpha:\begin{cases} 1 & \mapsto [1..|\gamma|] \\ k > 1 & \mapsto \emptyset \end{cases}$ is an embedding of $\gamma$ in $\pi$. Indeed $\gamma_{[1..|\gamma|]} = \gamma$ and $\pi\{\gamma, \varepsilon, \ldots, \varepsilon\} = \gamma$.

Note that if we denote the non-empty images of $\alpha$ by $\alpha^1, \ldots, \alpha^N$ and if we remove from $\pi$ the $\pi_i$ such that $\alpha(i) = \varepsilon$, we obtain a pattern $\sigma$ of $\pi$ such that $\gamma = \sigma[\gamma_{\alpha^1}, \ldots, \gamma_{\alpha^N}]$. But this pattern $\sigma$ may occur at several places in $\pi$ so a block decomposition $\gamma = \sigma[\gamma_{\alpha_1}, \ldots, \gamma_{\alpha_N}]$ may correspond to several embeddings of $\gamma$ in $\pi$.

**Example 5.6.** There are 12 embeddings of $\gamma = 546312$ into $\pi = 3142$, shown in Table 1. For instance, when writing $\gamma$ as the substitution $\circ[3241, 12]$, they are derived from the generalized substitutions $\pi\{3241, 12, \varepsilon, \varepsilon\}$, $\pi\{3241, \varepsilon, \varepsilon, 12\}$ and $\pi\{\varepsilon, \varepsilon, \varepsilon, 3241, 12\}$, corresponding to the three occurrences of $21$ in $\pi$. But when writing $\gamma$ as $312\{3241, 1, 1\}$, since $312$ has only one occurrence in $\pi$, only one embedding is derived, which comes from $\pi\{3241, 1, \varepsilon, 1\}$.

\(^7\)Recall that in this article, an interval of a permutation is a set of indices corresponding to a block of the permutation (see Definition 2.5 p.3).
Note that this definition of embeddings conveys the same notion as in [1], but it is formally different and it will turn out to be more adapted to the definition of the sets $E_{i,j}$ in Section 5.3.

For now, we present how to compute the embeddings of some permutation $\gamma$ into a permutation $\pi$. This is done with ALLEMBDINGS (Algo. 2) in two main steps, as suggested by the two parts of Table 1. First we compute all block decompositions of $\gamma$ (the left part of the table shows only some block decompositions of $\gamma = 5 4 6 3 1 2$, namely those that can be expressed as a generalized substitution in $\pi = 3 1 4 2$). Then for each block decomposition of $\gamma$, we compute all the embeddings of $\gamma$ into $\pi$ which correspond to this block decomposition (this is the right part of the table).

**Algorithm 2: ALLEMBDINGS($\gamma, \pi$)**

- **Data:** Two permutations $\gamma$ and $\pi$
- **Result:** The set of embeddings of $\gamma$ into $\pi$

```plaintext
begin
    $\mathcal{E} \leftarrow \emptyset$
    $\mathcal{D} \leftarrow \text{BLOCKDECOMPOSITIONS}(\gamma)$ /* See Algo. 3 */
    foreach $d \in \mathcal{D}$ do
        $\mathcal{E} \leftarrow \mathcal{E} \cup \text{EMBEDDINGS}(d, \pi)$ /* See Algo. 4 */
    return $\mathcal{E}$
end```

The procedure BLOCKDECOMPOSITIONS (Algo. 3) finds all the block decompositions of $\gamma$. It needs to compute all the intervals of $\gamma$. Such intervals can be described as pairs of indices $(i, j)$ with $i \leq j$ such that $\max_{i \leq k \leq j}(\gamma_k) - \min_{i \leq k \leq j}(\gamma_k) = j - i$. Denoting by $p$ the size of $\gamma$, this remark allows to compute easily all the intervals starting at $i$, for each $i \in [1..p]$ – see INTERVALS($\gamma, i$). Then, BLOCKDECOMPOSITIONS builds the set $\mathcal{D}$ of all sequences of intervals of the form $((i_1, j_1), \ldots, (i_m, j_m))$ with $i_1 = 1$, $j_m = p$, and $i_{k+1} = j_k + 1$ for all $k < m$. These sequences correspond exactly to the block decomposition.
Algorithm 3: BlockDecompositions(γ)

**Data:** A permutation γ

**Result:** The set D of block decompositions of γ

begin
  D ← ∅; P ← Intervals(γ, 1)
  foreach u = ((i_1, j_1), ..., (i_m, j_m)) ∈ P do
    P ← P \ {u}
    if j_m = |γ| then D ← D ∪ {u}
    else S ← Intervals(γ, j_m + 1)
    P ← P ∪ {u · s | s ∈ S}
  return D

/* Returns the set of intervals of γ of the form (i, j) */
Intervals(γ, i)
  I ← ∅
  max ← γ_0; min ← γ_i
  for j from i to |γ| do
    max ← max(max, γ_j); min ← min(min, γ_j)
    if max − min = j − i then I ← I ∪ {(i, j)}
  return I

decompositions of γ. They are computed iteratively, starting from all the intervals of the form (1, i) and examining how they can be extended to a sequence of intervals in D. To this end, note that any sequence ([(i_1, j_1), ..., (i_m, j_m)]) as above but such that j_m ≠ p is the prefix of at least one sequence of D, since for every i ∈ [1..p], at least (i, i) is an interval of γ.

For each i ∈ [1..p], computing Intervals(γ, i) is done in O(p), so O(p^2) is enough to compute once and for all the results of Intervals(γ, i) for all i ∈ [1..p] (provided we store them). The computation of all block decompositions of γ with BlockDecompositions(γ) then costs O(p^2p). Indeed there are at most 2^p−1 such sequences of D and each sequence of D contains at most p intervals. At each step, the algorithm extends a sequence already built. Thus the (amortized) overall complexity is O(p^2p) and this bound is tight.

The procedure Embeddings (Algo. 4) finds the embeddings of γ in π which correspond to a given block-decomposition of γ. The output D of BlockDecompositions corresponds to all the substitutions σ[γ(1), ..., γ(m)] which are equal to γ. For each d ∈ D we first determine the corresponding skeleton σ as the normalization of a sequence s_1...s_m of m integers, where each s_i is an element of γ falling into the i-th block of this decomposition. Then, we compute the set of occurrences of σ in π, since they are in one-to-one correspondence with embeddings of γ into π which corresponds to the given substitution γ = σ[γ(1), ..., γ(m)]. These occurrences are naively computed by testing, for every subsequence of m elements of π, whether it is an occurrence of σ.

The preprocessing part of Embeddings (Algo. 4), which consists in the computation
Algorithm 4: Embeddings($d, \pi$)

**Data:** A permutation $\pi$ of size $n$ and a block decomposition $d \in \mathcal{D}$ of $\gamma$,

$$d = ((i_1, j_1), \ldots, (i_m, j_m))$$

**Result:** The set of embeddings of $\gamma$ into $\pi$ which correspond to $d$

begin
  if $m > n$ then /* No embedding of $\gamma$ into $\pi$ corresponding to $d$ */
    return $\emptyset$
  else /* $\sigma$ is the skeleton such that $d = \sigma[\gamma(1), \ldots, \gamma(m)]$ */
    $\sigma \leftarrow \gamma(i_1, i_2, \ldots, i_m)$
    $E \leftarrow \emptyset$
    foreach subset $S = \{s_1, \ldots, s_m\}$ of $\{1, 2, \ldots, n\}$ with $s_1 < \ldots < s_m$ do
      if $\pi(s_1, s_2, \ldots, s_m) = \sigma$ then
        $\alpha \leftarrow$ embedding of $\gamma$ in $\pi$ such that
        $$\alpha(s_k) = [i_k..j_k] \text{ for } 1 \leq k \leq m$$
        $$\alpha(i) = \varepsilon \text{ for } i \notin S$$
        $E \leftarrow E \cup \{\alpha\}$
    end
  end
  return $E$
end

of the skeleton $\sigma$, costs $O(m)$. The examination of all possible occurrences of $\sigma$ in $\pi$ is then performed in $O(m n^m)$, where $n$ is the size of $\pi$.

In total, computing the set $\mathcal{D}$ of block decompositions of $\gamma$ is performed in $O(p 2^p)$. The set $\mathcal{D}$ contains at most $\binom{m}{m-1}$ block decompositions in $m$ blocks, so that the complexity of the main loop of ALLFINDS is at most $\sum_{m=1}^{p} m \binom{n}{m-1} (p-1) \leq p n^{2+p-1}$. Note that it can be reduced to $\min(p, n) 2^{n+\min(p,n)-1}$, by discarding all block decompositions of $\gamma$ in $m > n$ blocks, since these will never be used in an embedding of $\gamma$ into $\pi$. The upper bound $\sum_{m=1}^{p} m \binom{n}{m-1}$ is tight, as can be seen with $p \leq n$, $\gamma = 12 \ldots p$ and $\pi = 12 \ldots n$.

In the next section, we explain how to use embeddings to propagate the pattern avoidance constraints in the subtrees.

5.3. Propagating constraints

To compute our combinatorial system, we compute equations for sets $\hat{\mathcal{C}}^d(E)$ (initially for $\hat{\mathcal{C}}(B^*)$, that is $\mathcal{C}$) starting from Equation (1), (2) or (3) (p.14). For every set $\pi[\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_n]$ that appears on the right-hand side of the equation, we push the pattern avoidance constraints of $E$ in the subtrees. This is achieved using embeddings of excluded patterns in the root $\pi$. For instance, assume that $\gamma = 546312 \in B^*$ and $\mathcal{S}_C = \{3142\}$, and consider $3142[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \hat{\mathcal{C}}](\gamma)$. The embeddings of $\gamma$ in 3142 indicates how the pattern $\gamma$ can be found in the subtrees in 3142[$\hat{\mathcal{C}}, \hat{\mathcal{C}}, \hat{\mathcal{C}}$]. The first embedding of Example 5.6 indicates that the full pattern $\gamma$ can appear all included in the first subtree. On the other hand, the last embedding of the same example tells us that $\gamma$ can spread over all the subtrees of 3142 except the third one. In order to avoid this particular embedding of $\gamma$, it is enough to avoid one of the induced pattern $\gamma_I$ in one of the subtrees. However, in
order to ensure that \( \gamma \) is avoided, the constraints resulting from all the embeddings must be considered and merged. This is formalized in Proposition 5.7.

**Proposition 5.7.** Let \( \pi \) be a simple permutation of size \( n \) and \( C_1, \ldots, C_n \) be sets of permutations. For any permutation \( \gamma \), the set \( \pi[C_1, \ldots, C_n] \{\gamma\} \) can be described as a union of sets of the form \( \pi[D_1, \ldots, D_n] \) where, for all \( i \), \( D_i = C_i \{\gamma_i\} \) and the restrictions appearing after \( \gamma \) (if there are any) are patterns of \( \gamma \) corresponding to normalized blocks of \( \gamma \). More precisely, we have

\[
\pi[C_1, \ldots, C_n] \{\gamma\} = \bigcup_{(k_1, \ldots, k_\ell) \in K^\pi} \pi[C_1(\langle E_{1,k_1} \ldots k_\ell \rangle), \ldots, C_n(\langle E_{n,k_1} \ldots k_\ell \rangle)]
\]

where \( K^\pi_\gamma = \{(k_1, \ldots, k_\ell) \in [1..n]^\ell \mid \forall i, \; \gamma_{\alpha_i(k_i)} \neq \varepsilon \text{ and } \gamma_{\alpha_i(k_i)} \neq 1\} \) and \( E_{m,k_1 \ldots k_\ell} = \{\gamma_{\alpha_i(k_i)} \mid i \in [1..\ell] \text{ and } k_i = m\} \), \( \{\alpha_1, \ldots, \alpha_\ell\} \) being the set of embeddings of \( \gamma \) in \( \pi \).

Similarly, \( \oplus[C_1^+, C_2](\gamma) \) (resp. \( \ominus[C_1^-, C_2](\gamma) \) ) can be described as the union of sets \( \oplus[D_1^+, D_2] \) (resp. \( \ominus[D_1^-, D_2] \) ) with \( D_i = C_i(\langle E_{i,k_1} \ldots k_\ell \rangle) \).

Proposition 5.7 and its proof heavily borrow from the proof of Lemma 18 of [1]. There are however several differences.

In the first part of our statement, while we stop at the condition \( D_i = C_i(\gamma_i) \) (which appears in the proof of Lemma 18 in [1]), [1] rephrases it as: \( D_i = C_i(\gamma_i) \) or a strong subclass of this class (that is, a proper subclass of \( C_i \) which has the property that every basis element of \( D_i \) is involved in some basis element of \( C_i \)). Note that this rephrasing is not always correct (but this does not affect the correctness of the result of [1] which uses their Lemma 18). A counter example is obtained taking \( \pi = 2413, C_i = Av(12) \) for all \( i \) and \( \gamma = 2143 \). Then, because of the embedding of \( \gamma \) in \( \pi \) that maps 21 in \( \pi_1 \) and 43 in \( \pi_2 \), we get a term with \( D_1 = C_1(2143, 21) = Av(12, 21) \), which is not a strong subclass of \( C_1(\gamma) = Av(12) \).

The second part of Proposition 5.7 is not present in [1]. That is, Proposition 5.7 goes further than the proof of Lemma 18 of [1]: we provide a statement which is nonetheless constructive as the proof of [1], but also explicit, so that it can be directly used for algorithmic purpose.

**Proof.** We first consider the case of a simple root \( \pi \). Let \( \gamma \) be a permutation and \( \{\alpha_1, \ldots, \alpha_\ell\} \) be the set of embeddings of \( \gamma \) in \( \pi \), each \( \alpha_i \) being associated to the generalized substitution \( \gamma = \pi[\gamma_{\alpha_i(1)} \ldots, \gamma_{\alpha_i(n)}] \).

Let \( \sigma = \pi[\sigma^{(1)} \ldots, \sigma^{(n)}] \), where each \( \sigma^{(k)} \in C_k \). Then \( \sigma \) avoids \( \gamma \) if and only if for every embedding \( \alpha_i \) (with \( 1 \leq i \leq \ell \)), there exists \( k_i \in [1..n] \) such that \( \gamma_{\alpha_i(k_i)} \) is not a pattern of \( \sigma^{(k_i)} \), i.e. such that \( \sigma^{(k_i)} \) avoids \( \gamma_{\alpha_i(k_i)} \). Equivalently, \( \sigma \) avoids \( \gamma \) if and only if there exists a tuple \( (k_1, \ldots, k_\ell) \in [1..n]^\ell \) such that for every embedding \( \alpha_i \) (with \( 1 \leq i \leq \ell \)), \( \sigma^{(k_i)} \) avoids \( \gamma_{\alpha_i(k_i)} \). Thus

\[
\pi[C_1, \ldots, C_n] \{\gamma\} = \bigcup_{(k_1, \ldots, k_\ell) \in [1..n]^\ell} \bigcap_{i=1}^\ell \pi[C_{k_i}, \gamma_{\alpha_i(k_i)}] \ldots, C_n].
\]

But, for any set \( A \) of permutations we have \( A(\varepsilon) = \emptyset \). So if for some \( i \in [1..\ell], \gamma_{\alpha_i(k_i)} = \varepsilon \), then \( \bigcap_{i=1}^\ell \pi[C_{k_i}, \gamma_{\alpha_i(k_i)}] \ldots, C_n] = \emptyset \). The same goes for the trivial permutation.
for any sets of permutations $E_k$.

Example 5.8. For $\forall k_1, \ldots, k_\ell \in [1..\ell] \mid \forall i$, $\gamma_{\alpha_i(k_i)} \neq \epsilon$ and $\gamma_{\alpha_i(k_i)} \neq 1$. Following Example 5.6, and denoting $\alpha_1$ to $\alpha_2$ the embeddings of Table 1 from top to bottom, we have for example $k = (1, 2, 3, 4, 1, 4, 2, 1, 4, 1) \in K_5^\pi$.

Moreover, as $\pi$ is simple, by uniqueness of the substitution decomposition we have for any sets of permutations $E_1, \ldots, E_n, F_1, \ldots, F_n$:

$$\pi[C_1(E_1), \ldots, C_n(E_n)] \cap \pi[F_1, \ldots, F_n] = \pi[C_1(E_1 \cup F_1), \ldots, C_n(E_n \cup F_n)].$$

Thus,

$$\pi[C_1, \ldots, C_n] = \bigcup_{(k_1, \ldots, k_\ell) \in K_\pi} \pi[C_1(E_1, k_1 \ldots k_\ell), \ldots, C_n(E_n, k_1 \ldots k_\ell)].$$

where $E_{m,k_1 \ldots k_\ell} = \{\gamma_{\alpha_i(k_i)} \mid i \in [1..\ell] \text{ and } k_i = m\}$. Following the same example again, $E_{1,k} = \{\gamma_{\alpha_1(1)}, \gamma_{\alpha_3(1)}, \gamma_{\alpha_{12}(1)}\} = \{546312, 213, 3241\}$, $E_{2,k} = \{546312, 12\}$, $E_{3,k} = \{546312\}$ and $E_{1,k} = \{546312, 312, 12\}$.

Proposition 5.7 follows as soon as we ensure that each set $E_{m,k_1 \ldots k_\ell}$ contains $\gamma$ and possibly other restrictions that are patterns of $\gamma$ corresponding to normalized blocks of $\gamma$. For a given $m \in [1..n]$, there always exists an embedding of $\gamma$ in $\pi$ that maps the whole permutation $\gamma$ to $\pi_m$. Denoting this embedding $\alpha_{jm}$, we have $\gamma_{\alpha_{jm}(m)} = \gamma$ and $\gamma_{\alpha_{jm}(q)} = \epsilon$ for $q \neq m$. Consequently, if $k_m \neq m$ then $\gamma_{\alpha_{jm}(k_m)} = \epsilon$ and $(k_1, \ldots, k_\ell) \notin K_\pi$. Therefore when $(k_1, \ldots, k_\ell) \in K_\pi$ then $k_m = m$ and the set $E_{m,k_1 \ldots k_\ell}$ contains at least $\gamma_{\alpha_{jm}(k_m)} = \gamma$. Moreover, by definition its other elements are patterns of $\gamma$ corresponding to normalized blocks of $\gamma$.

Notice now that the proof immediately extends to the case of roots $\oplus$ and $\ominus$. Indeed, in the above proof, we need $\pi$ to be simple only because we use the uniqueness of the substitution decomposition. In the case $\pi = \ominus$ (resp. $\oplus$), the uniqueness is ensured by taking the set $C_\pi$ of $\ominus$-indecomposable permutations of $C_1$ (resp. the set $C_\pi$ of $\ominus$-indecomposable permutations).

Example 5.8. For $\pi = \ominus$ and $\gamma = 3412$, there are three embeddings of $\gamma$ in $\pi$: $\alpha_1$ which follows from the generalized substitution $3412 = \ominus\{3412, \epsilon\}$, $\alpha_2$ which follows from $3412 = \ominus\{\epsilon, 3412\}$, and $\alpha_3$ which follows from $3412 = \ominus\{12, 12\}$. So $K_\pi = \{(1, 2, 1), (1, 2, 2)\}$ and the application of Equation (8) gives

$$\ominus[C_1, C_2](3412) = \ominus[C_1(3412, 12), C_2(3412)] \cup \ominus[C_1(3412), C_2(3412, 12)],$$

which simplifies to $\ominus[C_1(12), C_2(3412)] = \ominus[C_1(12), C_2(3412)] \cup \ominus[C_1(3412), C_2(12)].$

By induction on the size of $P$, Proposition 5.7 extends to the case of a set $P$ of excluded patterns, instead of a single permutation $\gamma$.
Proposition 5.9. For any simple permutation \( \pi \) of size \( n \) and for any set of permutations \( P \), the set \( \pi[C_1, \ldots, C_n](P) \) can be described as a union of sets \( \pi[D_1, \ldots, D_n] \) where for all \( i \), \( D_i = C_i(P \cup P_i) \) with \( P_i \) containing only permutations corresponding to normalized blocks of elements of \( P \).

Similarly, \( \circ[C_1^+, C_2](P) \) (resp. \( \circ[C_1^-, C_2](P) \)) can be described as a union of sets \( \circ[D_1^+, D_2] \) (resp. \( \circ[D_1^-, D_2] \)) where for \( i = 1 \) or 2, \( D_i = C_i(P \cup P_i) \) with \( P_i \) containing only permutations corresponding to normalized blocks of elements of \( P \).

Now we have all the results we need to describe an algorithm computing a (possibly ambiguous) combinatorial system describing \( \mathcal{C} \).

5.4. An algorithm computing a combinatorial system describing \( \mathcal{C} \)

We describe the algorithm AMBIGUOUSSYSTEM (Algo. 1 p.16) that takes as input the basis \( B \) of a class \( \mathcal{C} \) and the set \( \hat{\mathcal{C}} \) of simple permutations in \( \mathcal{C} \) (both finite), and that produces in output a (possibly ambiguous) system of combinatorial equations describing the permutations of \( \mathcal{C} \) through their decomposition trees. Recall (from p.16) that \( B^* \) denotes the subset of non-simple permutations of \( B \).

The main step is performed by EqnForClass (Algo. 5 below), taking as input \( \mathcal{S}_C \), \( \delta \in \{\,+, -\} \) and a set \( E \) of patterns (initially \( B^* \)), and producing an equation describing \( \hat{\mathcal{C}}^\delta(E) \). The algorithm takes an equation of the form (1), (2) or (3) (p.14) describing \( \hat{\mathcal{C}}^\delta(E) \) and adds one by one the constraints of \( E \) using Equation (8) as described in procedure AddConstraints.

---

Algorithm 5: EqnForClass(\( \hat{\mathcal{C}}^\delta, E \))

**Data:** \( E \) is a set of permutations, \( \hat{\mathcal{C}}^\delta \) is given by \( \mathcal{S}_C \) and \( \delta \in \{\,+, -\} \)

**Result:** An equation defining \( \hat{\mathcal{C}}^\delta(E) \) as a union of \( \pi[C_1, \ldots, C_n] \)

```
begin
\( \mathcal{E} \leftarrow \text{Equation (1) or (2) or (3) p.14 (depending on } \delta, \text{ replacing the left part } \hat{\mathcal{C}}^\delta \text{ by } \hat{\mathcal{C}}^\delta(E) \)

foreach \( \gamma \text{ in } E \) do
    foreach \( t = \pi[C_1, \ldots, C_n] \text{ that appears in } \mathcal{E} \) do
        \( t \leftarrow \text{AddConstraints}(\pi[C_1, \ldots, C_n], \gamma) \) /* this step modifies } \mathcal{E} */
    return \( \mathcal{E} \)

/* Returns a rewriting of \( \pi[C_1 \ldots C_n](\gamma) \) as a union \( \bigcup \pi[D_1, \ldots, D_n] \) */
AddConstraints (\( \pi[C_1 \ldots C_n], \gamma \))
computation all the embeddings of } \gamma \text{ in } \pi \text{ with ALLEMBEDDINGS} /* See Algo. 2 (p.19) */
compute } K_\gamma \text{ and sets } E_{m,k_1 \ldots k_t} \text{ defined in Equation (8)
return } \bigcup_{(k_1, \ldots, k_t) \in K_\gamma} \pi[C_1(E_{1,k_1 \ldots k_t}), \ldots, C_n(E_{n,k_1 \ldots k_t})]
```

The procedure AMBIGUOUSSYSTEM keeps adding new equations to the system which consists originally of the equation describing \( \hat{\mathcal{C}}(B^*) \), that is \( \mathcal{C} \). This algorithm repeatedly calls EqnForClass until every \( \hat{\mathcal{C}}^\delta(E_i) \) appearing in the system is defined by an equation.
All the sets $E$ are sets of normalized blocks (therefore of patterns) of permutations in $B^*$. Since $B$ is finite, there is only a finite number of patterns of elements of $B^*$, hence a finite number of possible $E$, and AMBIGUOUSSystem terminates. As for its complexity, it depends on the number of equations in the output system for which we give bounds in Section 6.5.

The correctness of the algorithm is a consequence of Propositions 5.3, 5.7 and 5.9, which therefore completes the proof of Theorem 5.1.

**Example 5.10.** Consider the class $C = \text{Av}(B)$ for $B = \{1243, 2413, 531642, 41352\}$: $C$ contains only one simple permutation (namely $3142$), and $B^* = \{1243\}$. Applying the procedure AMBIGUOUSSystem to this class $C$ gives the following system of equations:

$$
\hat C(1243) = 1 \cup \oplus[\hat C^+(12), \hat C(132)] \cup \ominus[\hat C^+(1243), \hat C(21)] \cup \ominus[\hat C^-(1243), \hat C(1243)]
$$

$$
\cup 3142[\hat C(1243), \hat C(12), \hat C(21), \hat C(132)] \cup 3142[\hat C(12), \hat C(12), \hat C(132), \hat C(132)] (9)
$$

$$
\hat C(132) = 1 \cup \ominus[\hat C^-(132), \hat C(132)] (11)
$$

$$
\hat C(21) = 1 \cup \oplus[\hat C^+(21), \hat C(21)]. (12)
$$

$$
\hat C^+(1243) = \ldots
$$

This is a simplified version of the actual output of AMBIGUOUSSystem. For instance, with a literal application of the algorithm, instead of Equation (9) we would get:

$$
\hat C(1243) = 1 \cup \oplus[\hat C^+(1243, 12), \hat C(1243, 132)] \cup \ominus[\hat C^+(1243), \hat C(1243, 132, 21)] \cup \ominus[\hat C^-(1243), \hat C(1243)]
$$

$$
\cup 3142[\hat C(1243, 12), \hat C(1243, 12), \hat C(1243, 132), \hat C(1243, 132)] \cup 3142[\hat C(1243, 12), \hat C(1243, 12), \hat C(1243, 132), \hat C(1243, 132, 21)]
$$

$$
\cup 3142[\hat C(1243, 12), \hat C(1243, 12), \hat C(1243, 132, 21), \hat C(1243, 132)] \cup 3142[\hat C(1243, 12), \hat C(1243, 132, 21), \hat C(1243, 132)]
$$

$$
\cup 3142[\hat C(1243, 12), \hat C(1243, 132, 21), \hat C(1243, 132, 21)] \cup 3142[\hat C(1243, 12), \hat C(1243, 132, 21), \hat C(1243, 132, 21)]
$$

Nevertheless, this union can be simplified. The simplification process will be described more thoroughly in Section 6.6. We illustrate it by two examples. First, since a permutation that avoids 12 or 132 will necessarily avoid 1243, the term $\ominus[\hat C^+(1243, 12), \hat C(1243, 132)]$ can be written as $\ominus[\hat C^+(12), \hat C(132)]$ (see Proposition 6.16 p. 34). We can also remove some terms of the union, such as

$$
3142[\hat C(1243, 12), \hat C(1243, 12), \hat C(1243, 132), \hat C(1243, 132, 21)]
$$

which is included in $3142[\hat C(12), \hat C(12), \hat C(132), \hat C(132)]$ (see Proposition 6.18 p. 35). Such simplifications can be performed on the fly, each time a new equation is computed.

We observe on Example 5.10 that the system produced by AMBIGUOUSSystem (Algo. 1) is ambiguous in general. In this case, Equation (9) gives an ambiguous description of the class $\hat C(1243)$: the two terms with root $\ominus$ have non-empty intersection, and similarly for root $3142$. Following the route of [1], we could use inclusion-exclusion on this system. On Equation (9) of Example 5.10, this would give:

$$
\hat C(1243) = 1 \cup \oplus[\hat C^+(12), \hat C(132)] \cup \ominus[\hat C^+(1243), \hat C(21)] \cup \ominus[\hat C^+(12), \hat C(21)]
$$

$$
\cup 3142[\hat C(12), \hat C(12), \hat C(12), \hat C(132)] \cup 3142[\hat C(12), \hat C(12), \hat C(21), \hat C(132)] \cup 3142[\hat C(12), \hat C(12), \hat C(21), \hat C(132)].
$$

As explained earlier, this is not the route we follow. In the next section, we explain how to modify the algorithm AMBIGUOUSSystem to obtain a combinatorial specification by introducing pattern containment constraints.

25
6. A non-ambiguous combinatorial system, i.e., a combinatorial specification

The goal of this section is to describe an algorithm computing a specification for any permutation class having finitely many simple permutations — see algorithm Specification (Algo. 6). This algorithm proceeds as AmbiguousSystem (Algo. 1 p.16), but also transforms each equation produced into a non-ambiguous one. The disambiguation of equations is performed by Disambiguate (Algo. 8). This algorithm replaces ambiguous unions appearing in an equation by disjoint unions using complement sets, in the spirit of Figure 5.

\begin{align*}
A \cup B \cup C &= 1 \uplus 2 \uplus 3 \uplus 4 \uplus 5 \uplus 6 \uplus 7 \\
&= (A \cap B \cap C) \uplus (A \cap B \cap C) \uplus (A \cap B \cap C) \\
&\uplus (A \cap B \cap C) \uplus (A \cap B \cap C) \uplus (A \cap B \cap C) \\
&\uplus (A \cap B \cap C)
\end{align*}

Figure 5: Rewriting unions as disjoint unions.

This may result in new terms appearing on the right side of the modified equation. Indeed, the terms of the system obtained from AmbiguousSystem involve pattern avoidance constraints (which were denoted \((E)\)). Consequently, taking complements, we are left with new pattern containment constraints as well (which we will denote \((A)\)). These new terms need to be defined by an equation, to be added to the system. This is solved by the procedure EqnForRestriction (Algo. 7), whose working principle is similar to that of EqnForClass.

Finally, EqnForRestriction and Disambiguate combine into Specification, which is our main algorithm (Algo. 6).

\begin{algorithm}
\caption{Specification\((B,S_C)\)}
\begin{algorithmic}
\State \textbf{Data:} A finite basis of patterns defining \(C = Av(B)\) and the finite set \(S_C\) of simple permutations in \(C\).
\State \textbf{Result:} A combinatorial specification defining \(C\).
\State begin
\State \(E \leftarrow \text{EqnForRestriction}(\hat{C},B^*,\emptyset)\) \hfill \(*\text{ See Algo.} 7\) while there is an equation \(F\) in \(E\) that has not been processed do
\State \(F \leftarrow \text{Disambiguate}(F)\) \hfill \(*\text{ See Algo.} 8\) *
\State while there exists a right-only restriction \(\hat{C}^\delta(E)(A)\) in equation \(F\) do
\State \(E \leftarrow E \cup \text{EqnForRestriction}(\hat{C}^\delta,E,A)\).
\State return \(E\)
\State \end
\end{algorithmic}
\end{algorithm}

6.1. Presentation of the method

We start by introducing notation to deal with the pattern containment constraints.

26
Definition 6.1. For any set $\mathcal{P}$ of permutations, we define $\mathcal{P}(E)(A)$ as the set of permutations of $\mathcal{P}$ that avoid every pattern of $E$ and contain every pattern of $A$:

$$\mathcal{P}(E)(A) = \{ \sigma \in \mathcal{P} \mid \forall \pi \in E, \pi \not\subseteq \sigma \text{ and } \forall \pi \in A, \pi \subseteq \sigma \}. $$

When $\mathcal{P} = \hat{C}^\delta$ for $\delta \in \{-, +, -\}$ and $\mathcal{C}$ a permutation class, such a set is called a restriction.

We also denote $\mathcal{P}(E) = \mathcal{P}(\emptyset)(E)$.

Restrictions are a generalization of permutations classes. For $\mathcal{A} = \emptyset$, $\mathcal{C}(E)(\emptyset)$ is the standard permutation class $\mathcal{C}(E)$. Any permutation class can be written as $S(E)$ with $S$ the set of all permutations and $E$ a set of patterns that may be infinite. Likewise, any restriction can be written as $S(E)(A)$ with $E$ a set of patterns that may be infinite, but we can always choose a finite set for $A$: if $A$ is infinite, then $S(E)(A) = \emptyset$.

Remark 6.2. Our convention that no permutation class contains the empty permutation $\varepsilon$ implies that $\varepsilon \notin \hat{C}^\delta(E)(A)$, for any restriction $\hat{C}^\delta(E)(A)$. We can also make the assumption that $\varepsilon \notin E$ and $\varepsilon \notin A$, since $\hat{C}^\delta(\{\varepsilon\}) = \emptyset$ and $\hat{C}^\delta(E)(\{\varepsilon\}) = \hat{C}^\delta(E)(A)$ for any $E$ and $A$. Moreover, we assume that $\mathcal{A} \cap E$ is empty and that $\emptyset \notin E$, otherwise $\hat{C}^\delta(E)(A) = \emptyset$. We finally assume that $1 \notin A$ since $\hat{C}^\delta(E)(A) = \hat{C}^\delta(E)(A \setminus \{1\})$.

Restrictions are stable by intersection as $\mathcal{P}(E)(A) \cap \mathcal{P}(E')(A') = \mathcal{P}(E \cup E')(A \cup A')$.

Definition 6.3. A restriction term is a set of permutations $\pi[D_1, D_2, \ldots, D_n]$ for $\pi$ a simple permutation, or $\oplus[D_1^+, D_2]$ or $\ominus[D_1^-, D_2]$, where each $D_i$ is a restriction of the form $\hat{C}(E)(A)$.

By uniqueness of the substitution decomposition of a permutation (Theorem 2.11 p.4), restriction terms are stable by intersection and the intersection is performed component-wise for terms sharing the same root: $\pi[D_1, \ldots, D_n] \cap \pi[T_1, \ldots, T_n] = \pi[D_1 \cap T_1, \ldots, D_n \cap T_n]$.

Now we have all the notions we need to present the general structure of our main algorithm: SPECIFICATION (Algo. 6 above). Adapting to restrictions the ideas developed for classes in Section 5, we obtain a non-ambiguous equation for any restriction $\hat{C}^\delta(E)(A)$ for $\delta \in \{-, +, -\}$ within four steps (where $1_{(A)} = 1$ if $A = \emptyset$ and $0$ otherwise):

- **Step 1**, equation from the substitution decomposition of permutations:
  $$\hat{C}^\delta(E)(A) = 1_{(A)} \cup \ldots \cup \left( \bigcup_{\pi \in \mathcal{S}_C} \pi[\hat{C}, \ldots, \hat{C}](E)(A) \right)$$

- **Step 2**, propagation of the avoidance constraints:
  $$\hat{C}^\delta(E)(A) = 1_{(A)} \cup \ldots \cup \left( \bigcup_{\pi \in \mathcal{S}_C} \left( \bigcup_{i \in I} \pi[\hat{C}(F^i_1), \ldots, \hat{C}(F^i_n)](A) \right) \right)$$

- **Step 3**, propagation of the containment constraints:
  $$\hat{C}^\delta(E)(A) = 1_{(A)} \cup \ldots \cup \left( \bigcup_{\pi \in \mathcal{S}_C} \left( \bigcup_{i \in I} \pi[\hat{C}(E^i_1)(A^i_1), \ldots, \hat{C}(E^i_n)(A^i_n)] \right) \right)$$

- **Step 4**, disambiguation of the equation:
  $$\hat{C}^\delta(E)(A) = 1_{(A)} \cup \ldots \cup \left( \bigcup_{\pi \in \mathcal{S}_C} \left( \bigcup_{k \in K} \pi[\hat{C}(E^{i_k}_1)(A^{i_k}_1), \ldots, \hat{C}(E^{i_k}_n)(A^{i_k}_n)] \right) \right)$$
The algorithm **Specification** starts by computing a non-ambiguous equation for $\mathcal{C} = \hat{\mathcal{C}}(\mathcal{B}^*)(\emptyset)$, calling the procedures $\text{EqnForRestriction}$ (performing Steps 1 to 3 above – see details in Algo. 7 p.29) and $\text{Disambiguate}$ (performing Step 4 – see details in Algo. 8 p.30). The specification for $\mathcal{C}$ is completed by using the same method of four steps to obtain an equation for each $\hat{\mathcal{C}}(\mathcal{E}_k^\delta)(\mathcal{A}_k^\delta)$ appearing on the right side of the produced equation, and again recursively to obtain an equation for each restriction that appears in the system.

Even if we use it only to compute specifications for permutation classes (writing $\mathcal{C} = \hat{\mathcal{C}}(\mathcal{B}^*)(\emptyset)$), this algorithm allows more generally to obtain a specification for any restriction $\mathcal{C}(\mathcal{E})(\mathcal{A})$ such that $\mathcal{C}(\mathcal{E})$ has a finite number of simple permutations (implying that $\mathcal{E}$ can be chosen finite). Indeed we only have to replace $\text{EqnForRestriction}(\hat{\mathcal{C}}, \mathcal{B}, \emptyset)$ with $\text{EqnForRestriction}(\hat{\mathcal{C}}\delta, \mathcal{E}, \mathcal{A})$, where $\mathcal{C} = \mathcal{S}(\mathcal{E})$.

We prove in the next three subsections that the result of our main algorithm **Specification** has the following properties:

**Theorem 6.4.** Let $\mathcal{C}$ be a permutation class given by its finite basis $\mathcal{B}$ and whose set $\mathcal{S}_{\mathcal{C}}$ of simple permutations is finite and given. Denote by $\mathcal{B}^*$ the set of non-simple permutations of $\mathcal{B}$.

The result of **Specification**$(\mathcal{B}, \mathcal{S}_{\mathcal{C}})$ (Algo. 6) is a finite combinatorial system of equations of the form $D_0 = 1[1 \in D_0] \uplus \bigoplus \pi[D_1, \ldots, D_n]$, where $1[1 \in D_0]$ is the class consisting in a unique permutation of size 1 if the permutation 1 belongs to $D_0$ and is empty otherwise, and where each $D_i$ is a restriction $\hat{\mathcal{C}}(\mathcal{E})(\mathcal{A})$ with $\delta \in \{ , +, - \}$, and $\mathcal{E}$ and $\mathcal{A}$ containing only normalized blocks\(^8\) of elements of $\mathcal{B}^*$. Moreover $\mathcal{C}$ appears as the left part of an equation, and every $D_i$ that appears in the system is the left part of one equation of the system.

In particular this provides a combinatorial specification of $\mathcal{C}$.

**Corollary 6.5.** Let $\mathcal{E}$ be the system of equations output by **Specification**$(\mathcal{B}, \mathcal{S}_{\mathcal{C}})$. Then the set $\{D : D$ is the left part of some equation of $\mathcal{E}\}$ is a query-complete set containing $\mathcal{C}$.

**Proof.** This a direct consequence of the form of $\mathcal{E}$ described in Theorem 6.4. \hfill \Box

6.2. Computing an equation for each restriction

Let $\hat{\mathcal{C}}(\mathcal{E})(\mathcal{A})$ be a restriction. Our goal here is to find an equation describing this restriction using smaller restriction terms (smaller with respect to inclusion).

If $\mathcal{A} = \emptyset$, this is exactly the problem addressed in Section 5 and solved by pushing down the pattern avoidance constraints with the procedure $\text{AddConstraints}$ of $\text{EqnForClass}$ (Algo. 5). The procedure $\text{EqnForRestriction}$ (Algo. 7) below shows how to propagate also the pattern containment constraints induced by $\mathcal{A} \neq \emptyset$.

The pattern containment constraints are propageted by $\text{AddMandatory}$, in a very similar fashion to the pattern avoidance constraints propagated by $\text{AddConstraints}$. To compute $t(\gamma)$ for $\gamma$ a permutation and $t = [\mathcal{D}_1, \ldots, \mathcal{D}_n]$ a restriction term, we first compute all embeddings of $\gamma$ into $\pi$. In this case, a permutation belongs to $t(\gamma)$ if and

---

\(^8\)Recall from Definition 2.6 (p.3) that the normalized blocks of $\gamma$ are the permutations $\gamma_I$ obtained when restricting $\gamma$ to any of its intervals $I$. 28
Algorithm 7: EqnForRestriction($\hat{C}_\delta, E, A$)

**Data:** $\hat{C}_\delta, E, A$ with $E, A$ sets of permutations, $\hat{C}_\delta$ given by $S_C$ and $\delta \in \{-, +, .\}$.

**Result:** An equation defining $\hat{C}_\delta(E)(A)$ as a union of restriction terms.

begin
\[ E \leftarrow \text{Equation (1) or (2) or (3) (depending on } \delta, \text{ replacing the left part } \hat{C}_\delta \text{ by } \hat{C}_\delta(E)(A) \]

foreach avoidance constraint $\gamma$ in $E$ do
\[
\begin{array}{l}
\text{foreach } t = \pi[C_1, \ldots, C_n] \text{ that appears in } E \text{ do} \\
\quad t \leftarrow \text{AddConstraints}(\pi[C_1, \ldots, C_n], \gamma) \quad */ \text{See Algo. 5 (p.24) */}
\end{array}
\]

foreach containment constraint $\gamma$ in $A$ do
\[
\begin{array}{l}
\text{foreach } t = \pi[C_1, \ldots, C_n] \text{ that appears in } E \text{ do} \\
\quad t \leftarrow \text{AddMandatory}(\pi[C_1, \ldots, C_n], \gamma)
\end{array}
\]

return $E$

**Proposition 6.6.** Let $\pi$ be a permutation of size $n$ and $D_1, \ldots, D_n$ be sets of permutations. For any permutation $\gamma$, let $\{\alpha_1, \ldots, \alpha_t\}$ be the set of embeddings of $\gamma$ in $\pi$, then
\[
\pi[D_1, \ldots, D_n](\gamma) = \bigcup_{i=1}^{t} \pi[D_1(\gamma(1)), D_2(\gamma(2)), \ldots, D_n(\gamma(n))]. \tag{13}
\]

For instance, for $t = 3142[1, 2, 3, 4]$ and $\gamma = 546312$, there are 12 embeddings of $\gamma$ into 3142 (see Table 1 p.19), and the embedding 3142[1, 2, 3, 4] contributes to the above union with the term 3142[1, 2, 3, 4] of the embedding 3142[1, 2, 3, 4] contributes to the above union with the term 3142[1, 2, 3, 4] of the embedding 3142[1, 2, 3, 4] contributes to the above union with the term 3142[1, 2, 3, 4].

**Proof.** Let $\sigma \in \pi[D_1, \ldots, D_n]$; then $\sigma = \pi[\sigma_1, \ldots, \sigma_n]$ where each $\sigma_k \in D_k$. Then $\sigma$ contains $\gamma$, if and only if there exists an embedding $\alpha_i$ of $\gamma = \pi[\gamma_1, \ldots, \gamma_n]$ such that $\gamma_\alpha = \sigma_j$ is a pattern of $\sigma_j$ for all $j$. \hfill \square

Hence, any restriction term $t = \pi[D_1, \ldots, D_n](\gamma)$ can be described as a (possibly ambiguous) union of restriction terms.

6.3. Disambiguation procedure

As explained earlier, Disambiguate (Algo. 8) disambiguates equations introducing complement sets.
Algorithm 8: Disambiguate($F$)

**Data:** A potentially ambiguous equation $F$ defining a restriction

**Result:** A non-ambiguous equation equivalent to $F$

```
begin
  foreach root $\pi$ that appears several times in $F$ do
    Replace the union of the restriction terms of $F$ whose root is $\pi$ by a disjoint union using Equations (14), (15) and (16) below.
  return $F$
end
```

Every equation produced by EqnForRestriction is written as $t = 1 \cup t_1 \cup t_2 \cup t_3 \cup \ldots \cup t_k$ where the sets $t_i$ are restriction terms (some $\pi[D_1, D_2, \ldots, D_n]$) and $t$ is a restriction (some $\hat{C}^\delta(E)A$). By uniqueness of the substitution decomposition of a permutation, restriction terms of this union which have different roots $\pi$ are disjoint. Thus for an equation we only need to disambiguate unions of terms with same root. For example in Equation (9) (p.25), there are two pairs of ambiguous terms which are terms with root 3142 and terms with root $\oplus$. Every ambiguous union can be written in an unambiguous way:

**Proposition 6.7.** Let $A_1, \ldots, A_n$ be $n$ sets and for each of them denote $\overline{A_i}$ the complement of $A_i$ in any set containing $\bigcup_{i=1}^{n} A_i$. The union $\bigcup_{i=1}^{n} A_i$ can be written as the disjoint union of the $2^n - 1$ sets of the form $\bigcap_{i=1}^{n} X_i$ with $X_i \in \{A_i, \overline{A_i}\}$ and at least one $X_i$ is equal to $A_i$.

This proposition is the starting point of the disambiguation. See Figure 5 (p.26) for an example. In order to use Proposition 6.7, we have to choose in which set we take complements.

**Definition 6.8.** For any restriction term $t$, we define its complement $\bar{t}$ as follows:

- if $t = \pi[D_1, \ldots, D_n]$ with $\pi$ simple and for all $i$, $D_i \subset \hat{C}$, we set $\bar{t} = \pi[\hat{C}, \ldots, \hat{C}] \setminus t$;
- if $t = \oplus[D_1, D_2]$ with $D_1, D_2 \subset \hat{C}$, we set $\bar{t} = \oplus[\hat{C}^+, \hat{C}] \setminus t$;
- if $t = \ominus[D_1, D_2]$ with $D_1, D_2 \subset \hat{C}$, we set $\bar{t} = \ominus[\hat{C}^-, \hat{C}] \setminus t$.

Moreover for any restriction $D$ of $\hat{C}^\delta$ with $\delta \in \{+, -, \}$, we set $\bar{D} = \hat{C}^\delta \setminus D$.

From Proposition 6.7, every ambiguous union of restriction terms sharing the same root in an equation of our system can be written in the following unambiguous way:

$$
\bigcup_{i=1}^{k} t_i = \bigcup_{X \subseteq \{1 \ldots k\}, X \neq \emptyset} t_i \cap \bigcap_{i \in X} \overline{t_i}.
$$

(14)

For instance, consider terms with root 3142 in Equation (9): $t_1 = 3142[\hat{C}(12), \hat{C}(12), \hat{C}(132), \hat{C}(132)]$ and $t_2 = 3142[\hat{C}(1243), \hat{C}(12), \hat{C}(21), \hat{C}(132)]$. Equation (14) applied to $t_1 \cup t_2$ in Equation (9) gives an expression of the form

$$
\hat{C}(1243) = 1 \cup \oplus[\ldots] \cup \ominus[\ldots] \cup \ominus[\ldots] \cup (t_1 \cap t_2) \cup (t_1 \cap \bar{t}_2) \cup (\bar{t}_1 \cap t_2).
$$

We now explain how to compute the complement $\bar{t}$ of a restriction term $t$. 30
Proposition 6.9. Let $t = \pi[D_1, \ldots, D_n]$ be a restriction term. Then $\bar{t}$ is the disjoint union of the $2^n - 1$ sets of the form $\pi[X_1, \ldots, X_n]$ with $X_i \in \{D_i, D'_i\}$, and not all $X_i$ are equal to $D_i$:

$$\bar{t} = \biguplus_{X_i \subseteq \{1, \ldots, n\}, X_i \neq \emptyset} \pi[D'_1, \ldots, D'_n] \text{ where } D'_i = D_i \text{ if } i \in X \text{ and } D'_i = D_i \text{ otherwise},$$

(15)

For example, $\bar{D_1, D_2} = \bar{D_1} \cup \bar{D_1, D_2} \cup \bar{D_1, D_2} \cup \bar{D_1, D_2}$.

Proof. Recall that $\bar{t} = \pi[C, \ldots, \bar{C}] \setminus t$. Let $\sigma = \pi[\sigma_1, \ldots, \sigma_n] \in \bar{t}$. Assume that for each $i$, $\sigma_i \in D_i$, by uniqueness of substitution decomposition we get a contradiction. Therefore $\sigma \in \pi[X_1, \ldots, X_n]$ with $X_i \in \{D_i, D'_i\}$, and not all $X_i$ are equal to $D_i$.

Conversely if $\sigma \in \pi[X_1, \ldots, X_n]$ with $X_i \in \{D_i, D'_i\}$ and at least one $X_i$ is equal to $D_i$, then $\sigma \in \pi[C, \ldots, \bar{C}]$ and $\sigma \notin t$.

Finally for such sets $X_i$ that are distinct, the sets $\pi[X_1, \ldots, X_n]$ are disjoint. Indeed $D_i \cap D'_i$ is empty and the writing as $\pi[\sigma_1, \ldots, \sigma_n]$ is unique. So the union describing $\bar{t}$ is disjoint. The proof is similar when $\pi = \emptyset$ or $\pi = \emptyset$.

Proposition 6.9 shows that $\bar{t}$ is not a restriction term in general. However it can be expressed as a disjoint union of some $\pi[D'_1, \ldots, D'_n]$, where the $D'_i$ are either restrictions or complements of restrictions. The complement operation being pushed from restriction terms down to restrictions, we now compute $\bar{D}$, for a given restriction $D = \bar{C}^\ell(E)(A)$, $\bar{D}$ denoting the set of permutations of $\bar{C}^\ell$ that are not in $D$. Note that, given a permutation $\sigma$ of $A$, then any permutation $\tau$ of $\bar{C}^\ell(\sigma)$ is in $\bar{D}$ because $\tau$ avoids $\sigma$ whereas permutations of $D$ must contain $\sigma$. Symmetrically, if a permutation $\sigma$ is in $E$ then permutations of $\bar{C}^\ell(\sigma)$ are in $\bar{D}$. It is straightforward to check that $\bar{C}^\ell(E)(A) = \bigcup_{\sigma \in E} \bar{C}^\ell(\sigma) \bigcup \bigcup_{\sigma \in A} \bar{C}^\ell(\sigma)$. Unfortunately this expression is ambiguous. As before, we can rewrite it as an unambiguous union:

Proposition 6.10. Let $D = \bar{C}^\ell(E)(A)$ be a restriction with $\delta \in \{+, -, \} \ k = |E|$ and $\ell = |A|$. Then $\bar{D}$ is the disjoint union of the $2^{k+\ell} - 1$ restrictions $\bar{C}^\ell(E')(A')$ with $(E', A')$ a partition of $E \cup A$ such that $(E', A') \neq (E, A)$. In other words,

$$\bar{C}^\ell(E)(A) = \biguplus_{X \subseteq A, Y \subseteq E, X \times Y \neq \emptyset} \bar{C}^\ell(X \cup \overline{Y})(Y \cup \overline{X}),$$

(16)

Proof. Let $\tau \in \bar{D} = \bar{C}^\ell \setminus D$. Define $E' = \{\pi \in E \cup A | \pi \notin \tau\}$ and $A' = \{\pi \in E \cup A | \pi \in \tau\}$. Observe that $\tau \in \bar{C}^\ell(E')(A')$. Moreover $E' \cap A' = \emptyset$, $E' \cup A' = E \cup A$ and $(E', A') \neq (E, A)$ otherwise $\tau$ would be in $D$.

Conversely let $\tau \in \bar{C}^\ell(E')(A')$ with $(E', A')$ a partition of $E \cup A$ such that $(E', A') \neq (E, A)$, then $\tau \in \bar{C}^\ell$. As $(E', A') \neq (E, A)$, either there is some $\sigma$ in $E$ that $\tau$ contains, or there is some $\sigma$ in $A$ that $\tau$ avoids. In both cases, $\tau \notin D$ thus $\tau \in \bar{C}^\ell \setminus D = \bar{D}$.

Finally for distinct partitions $(E', A')$ of $E \cup A$, the sets $\bar{C}^\ell(E')(A')$ are disjoints. Indeed a permutation in two sets of this form would have to both avoid and contain some permutation of $E \cup A$, which is impossible. □

31
Therefore using Equations (15) and (16) we have that for any restriction term \( C \), its complement \( \overline{C} \) can be expressed as a disjoint union of restriction terms:

\[
\overline{C} = \overline{(123)}(231, 123)(4321) = \overline{(123)}(231, 123)(4321) \cup \overline{(231)}(123, 4321) \cup \overline{(123)}(231, 4321) \cup \overline{(123)}(123, 231).
\]

Moreover by uniqueness of the substitution decomposition,

\[
\pi[D_1, \ldots, D_k \sqcup D_k', \ldots, D_n] = \pi[D_1, \ldots, D_k, \ldots, D_n] \sqcup \pi[D_1, \ldots, D_k', \ldots, D_n].
\]

Therefore using Equations (15) and (16) we have that for any restriction term \( t_i \), its disjoint union of restriction terms:

**Proposition 6.11.** For any restriction term \( t \), its complement \( \overline{t} \) can be written as a disjoint union of restriction terms. More precisely if \( t = \pi[D_1, \ldots, D_n] \) with \( D_i = \hat{C}^h(E_i\langle A_i \rangle) \) and \( m = \sum_{i=1}^{n} |E_i| + |A_i| \), then \( \overline{t} \) is the disjoint union of the \( 2^m - 1 \) restriction terms \( t = \pi[D_1', \ldots, D_n'] \) such that for all \( i \), \( D_i' = \hat{C}^h(E_i\langle A_i' \rangle) \) with \( (E_i', A_i') \) a partition of \( E_i \sqcup A_i \), and there exists \( i \) such that \( (E_i', A_i') \neq (E_i, A_i) \).

By distributivity of intersection over disjoint union, Equation (14) above can therefore be rewritten as a disjoint union of intersection of restriction terms. Because restriction terms are stable by intersection, the right-hand side of Equation (14) is hereby written as a disjoint union of restriction terms. This leads to the following result:

**Proposition 6.12.** Any union of restriction terms can be written as a disjoint union of restriction terms, and this can be done algorithmically using Equations (14), (15) and (16).

Altogether, for any equation of our system, we are able to rewrite it unambiguously with disjoint unions of restriction terms, using the algorithm **Disambiguate**.

### 6.4. Specification: an algorithm computing a combinatorial specification describing \( C \)

The procedures described above finally combine into **Specification** (Algo. 6 p.26), the main algorithm of this article, which computes a combinatorial specification for \( C \). Equations of the specification are computed iteratively, starting from the one for \( C \): this is achieved using **EqnForRestriction** (Algo. 7) described in Section 6.2, which produces equations that may be ambiguous. As we do not know how to decide whether an equation is ambiguous or not, we apply **Disambiguate** (Algo. 8) to every equation produced. Since some new right-only restrictions may appear during this process, to obtain a complete system we compute iteratively equations defining these new restrictions using again **EqnForRestriction**.

The termination of **Specification** is easy to prove. Indeed, for all the restrictions \( \hat{C}^h(E\langle A \rangle) \) that are considered in the inner while loop of **Specification**, every permutation in the sets \( E \) and \( A \) is a pattern of some element of the basis \( B \) of \( C \). And since \( B \) is finite, there is a finite number of such restrictions. Consequently, the algorithm produces an unambiguous system (i.e. a combinatorial specification) which is the result of a finite number of iterations of computing equations followed by their disambiguation.
6.5. Size of the specification obtained

The complexity of Specification (Algo. 6) depends on the number of equations in the computed specification, which may be quite large.\footnote{The same goes for the complexity of AmbiguousSystem. The bounds given in Proposition 6.13, Corollary 6.14 and Proposition 6.15 also hold for the number of equations of the system output by AmbiguousSystem.} We were not able to determine exactly how big it can be, and could only provide in Proposition 6.13 and Corollary 6.14 upper bounds on its size (i.e., number of equations) which seems to be (very) overestimated. We leave open the question of improving the upper bound on the size of the specification produced by our method. However, we point out that such an upper bound cannot be less than an exponential (in the sum of the sizes of the excluded patterns). Indeed, we give in Proposition 6.15 a generic example where our method produces such an exponential number of equations in the specification. Nevertheless, this convoluted example was created on purpose, and in many cases the number of equations obtained is not so high.

**Proposition 6.13.** Let $C = \text{Av}(B)$ and $B^*$ be the set of non-simple permutations of $B$. Let $P^*$ be the set of normalized blocks of permutations of $B^*$. The number of equations in the specification of $C$ computed by the procedure Specification is at most $3|P^*|$.  

*Proof.* In the specification we obtain, every equation is of the form  

$$\hat{C}\delta(E)(A) = \ldots,$$

where $\delta \in \{-, +, -\}$, $E \cup A \subseteq P^*$.  

As explained in Remark 6.2 (p.27), we can further assume that $E \cup A \subseteq P^* \setminus \{1\}$ and that $E \cap A = \emptyset$. The number of equations is then bounded by the number of such triplets $(\delta, E, A)$ which is $3^{1+|P^*|-1}$. \quad $\Box$

Using the previous proposition and the fact that the number of blocks of a permutation of size $k$ is less than $k^2$, we have the following consequence:

**Corollary 6.14.** Let $C = \text{Av}(B)$ and $t = \sum_{\pi \in B} |\pi|$. The number of equations in the specification of $C$ computed by Specification is at most $3^t$.  

However, Specification is designed to compute only the equations we need, and the number of equations produced is in practice much smaller. See for instance the example of Section 7, where $B^* = \{1243, 2341\}$ and $P^* = \{1, 12, 21, 123, 132, 1243, 2341\}$: the upper bound of Proposition 6.13 is $3^7 = 2187$, but only 16 equations are effectively computed. But as shown by the following proposition, Specification produces in the worst case a specification with a number of equations that is exponential in $t$.

**Proposition 6.15.** For each $n \geq 4$, there exists a class $C_n = \text{Av}(B_n)$ whose specification computed by the procedure Specification (Algo. 6) has at least $2^s$ equations, where the sum $t$ of the sizes of the elements of $B_n$ is approximately $s$, in the sense that $1 < t/s \ll s$.

*Proof.* For any $n \geq 4$, denote by $S_n$ the set of simple permutations of size $n$ and by $s_n$ its cardinality. Remember from [4] that $s_n \sim \frac{n!}{n^n}$. Fix some $n \geq 4$, let $s = s_{2n-1}$, and define $\gamma = \oplus[\tau_1, \ldots, \tau_s]$ with $\{\tau_i : 1 \leq i \leq s\} = S_{2n-1}$ and $B_n = S_{2n} \cup \{\gamma\}$. Note that $B_n$ is an
antichain, and consider the class $C_n = Av(B_n)$. The sum of the sizes of the elements of $B_n$ is $t = 2n \cdot s_{2n} + (2n - 1) \cdot s_{2n-1}$. Thus $s < t$ and $t/s \sim 4n^2$. Using Stirling formula, $t$ and $s$ are both of order $(2n)^{2n+\text{constant}}$.

It is not hard to see that $C_n$ contains a finite number of simple permutations. Indeed, it follows from [29] (see details in [27] for instance) that $C_n$ contains no simple permutations of size $2n$ or more. Note also that the simple permutation $\pi = 246135$ is small enough that it belongs to $C_n$. Moreover, $B_n^*_n = \{ \gamma \}$ and from Proposition 5.3, $C_n = \hat{C}_n(\gamma)$.

We claim that the computations performed by AddConstraints($\pi[\hat{C}_n, \ldots, \hat{C}_n], \gamma$) in EqnForRestriction($B_n$) (Algo. 7 p.29) will create at least $2^s$ right-only terms, thus giving rise to at least $2^s$ additional equations in the specification of $C_n$. More precisely, with notation from Proposition 5.7 (p.22), we prove that for each subset $E$ of $S_{2n-1}$, there exists a tuple $(k_1, \ldots, k_s) \in K^*_\gamma$ such that $E_{2,k_1,\ldots,k_s} = E \cup \{ \gamma \}$, ensuring that $\hat{C}_n(E \cup \{ \gamma \})$ appear in the system of equations.

Let us start by classifying the embeddings of $\gamma$ in $\pi$ into three categories. For all $i \in [1..s]$, let us denote by $\alpha_i$ the embedding of $\gamma$ in $\pi$ that sends $\tau_i$ to $\pi_2$, $\tau_{i} \uplus \tau_1, \ldots, \tau_{i-1}$ to $\pi_1$ and $\tau_i \uplus \tau_{i+1}, \ldots, \tau_s$ to $\pi_3$; and let $\alpha_{s+1}$ be the embedding of $\gamma$ in $\pi$ that sends $\gamma$ to $\pi_2$.

Note that for any $i \neq s+1$, there exists some $k_i \in [1..6]$, with $k_i \neq 2$, such that $\gamma_{\alpha_i(k_i)} \neq \epsilon$ and $\gamma_{\alpha_i(k_i)} \neq 1$ (since for each $j$, $\tau_j$ is simple and $|\pi| < |\tau_j|$, thus $\tau_j$ has to be entirely embedded in some $\pi_k$). This remark allows to consider, for each subset $E$ of $S_{2n-1}$, a tuple $(k_1, \ldots, k_s)$ defined as follows. For $i \in [1..s]$, we set $k_i = 2$ if $\tau_i \in E$, and otherwise we choose $k_i \neq 2$ such that $\gamma_{\alpha_i(k_i)} \neq \epsilon$ and $\gamma_{\alpha_i(k_i)} \neq 1$. We set $k_{s+1} = 2$. For $i \in [s+2..l]$, we choose $k_i \neq 2$ such that $\gamma_{\alpha_i(k_i)} \neq \epsilon$ and $\gamma_{\alpha_i(k_i)} \neq 1$. Consequently, the following properties hold:

- $(k_1, \ldots, k_s) \in K^*_\gamma$ (defined in Equation (8) p.22);
- $k_i = 2$ if and only if $\tau_i \in E$ or $i = s + 1$;
- for $i \in E$, $\gamma_{\alpha_i(k_i)} = \tau_i$;
- and for $i = s + 1$, $\gamma_{\alpha_i(k_i)} = \gamma$.

This ensures that $E_{2,k_1,\ldots,k_s} = E \cup \{ \gamma \}$ as claimed. \qed

6.6. Simplifications on the fly

During the computation of the equations by Specification (Algo. 6), many restriction terms appear, that may be redundant or a bit more intricate than necessary. For instance, in the equations obtained when pushing down the constraints in the subtrees using the rewriting described in Propositions 5.7 (p.22) and 6.6 (p.29), some element of a given union may not be useful because it may be included in some other element of the union. We simplify these unions by deleting useless elements, using Proposition 6.18 below. Moreover, when a restriction $D$ of the form $\hat{C}_n(E)(A)$ arise, it can often be written as $D = \hat{C}_n(E')(A')$ with $E'$ (resp. $A'$) having fewer elements than $E$ (resp. $A$). We use the description having as few elements as possible, thanks to Proposition 6.16. Proposition 6.17 further allows some trivial simplifications.

For any sets $E$ and $A$ of patterns, $\min E$ (resp. $\max A$) denote the subset of $E$ (resp. $A$) containing all minimal (resp. maximal) elements of $E$ (resp. $A$) for the pattern order $\preceq$.

**Proposition 6.16.** In any equation, every restriction $\hat{C}_n(E)(A)$ may be replaced by $\hat{C}_n(\min E')(\max A)$ without modifying the set of permutations described.
Proof. The proposition is an immediate consequence of $\hat{\mathcal{C}}^\delta(E)(A) = \hat{\mathcal{C}}^\delta(\min(E))(\max(A))$. This identity follows easily from two simple facts: if a permutation $\sigma$ avoids $\pi$ then $\sigma$ avoids all patterns containing $\pi$; and if $\sigma$ contains $\pi$ then $\sigma$ contains all patterns contained in $\pi$. \hfill $\square$

Proposition 6.17. The restriction term $\pi[C_1, \ldots, C_n]$ is the empty set if and only if there exists $i$ such that $C_i = \emptyset$. In this case, $\pi[C_1, \ldots, C_n]$ may be removed from any union of restriction terms without modifying the set of permutations described by this union.

Proposition 6.18. Consider a union of restriction terms containing two terms with the same root $\pi[C_1, \ldots, C_n]$ and $\pi[D_1, \ldots, D_n]$. If for all $i$, $C_i \subseteq D_i$, then we may remove $\pi[C_1, \ldots, C_n]$ from this union without modifying the set of permutations it describes.

Proof. If for all $i$, $C_i \subseteq D_i$, then $\pi[C_1, \ldots, C_n] \subseteq \pi[D_1, \ldots, D_n]$, giving the result immediately. \hfill $\square$

Performing the simplifications of Proposition 6.16 requires that we can compute $\min(E)$ and $\max(A)$ effectively. This can be done naively by simply checking all pattern relations between all pairs of elements of $E$ (resp. $A$). Similarly, to perform all simplifications induced by Propositions 6.17 and 6.18, we would need to be able to decide whether $C_i$ is empty or $C_i \subseteq D_i$, for any restrictions $C_i$ and $D_i$. Lemmas 6.19 and 6.20 below give sufficient conditions for the emptiness and the inclusion of a restriction into another, hence allowing to perform some simplifications.

Lemma 6.19. A restriction $\hat{\mathcal{C}}^\delta(E)(A)$ is the empty set as soon as there exist $\pi \in E$ and $\sigma \in A$ such that $\pi \preceq \sigma$.

Lemma 6.20. Let $\delta \in \{-, +, -\}$ and $E_1, E_2, A_1$ and $A_2$ be any sets of permutations. Then $\hat{\mathcal{C}}^\delta(E_1)(A_1) \subseteq \hat{\mathcal{C}}^\delta(E_2)(A_2)$ as soon as:

- for each $\pi \in E_2$, there exists $\tau \in E_1$ such that $\tau \preceq \pi$, and
- for each $\pi \in A_2$, there exists $\tau \in A_1$ such that $\pi \preceq \tau$.

Proof. Assume the two conditions of the statement are satisfied. Consider $\sigma \in \hat{\mathcal{C}}^\delta(E_1)(A_1)$, and let us prove that $\sigma \in \hat{\mathcal{C}}^\delta(E_2)(A_2)$.

Let $\pi \in E_2$. By assumption, there exists $\tau \in E_1$ such that $\tau \preceq \pi$. Since $\sigma \in \hat{\mathcal{C}}^\delta(E_1)$, $\sigma$ avoids $\tau$. From $\tau \preceq \pi$, we conclude that $\sigma$ also avoids $\pi$. Therefore $\sigma \in \hat{\mathcal{C}}^\delta(E_2)$.

Similarly, let $\pi \in A_2$. There exists $\tau \in A_1$ such that $\pi \preceq \tau$. Since $\sigma \in \hat{\mathcal{C}}^\delta(E_1)(A_1)$, we know that $\sigma$ contains $\tau$. Consequently, $\sigma$ also contains $\pi$. Hence $\sigma \in \hat{\mathcal{C}}^\delta(E_2)(A_2)$. \hfill $\square$

Note that the condition given by Lemma 6.20 is sufficient, but it is not necessary. Indeed for $E_1 = \{34152\}$, $A_1 = \{364152\}$, we have $\hat{\mathcal{C}}^\delta(E_1)(A_1) = \emptyset$. In particular $\hat{\mathcal{C}}^\delta(E_1)(A_1) \not\subseteq \hat{\mathcal{C}}^\delta(123)(132)$ even though the first condition of Lemma 6.20 is not satisfied. We may however wonder if the condition of Lemma 6.20 could be necessary under the assumption that $E_1$, $E_2$, $A_1$ and $A_2$ satisfy additional conditions, like $\hat{\mathcal{C}}^\delta(E_1)(A_1)$ being non-empty and $E_2$ and $A_2$ being antichains.

Similarly, the condition given by Lemma 6.19 is also sufficient, but it is not necessary. Indeed $\hat{\mathcal{C}}^\delta(132, 213, 231, 312)(12, 21)$ is empty even though the condition of Lemma 6.19 is not satisfied.
Any of the simplifications explained above can be performed on the fly, each time a new restriction term appears while running the procedure SPECIFICATION. This allows to compute systems of equations significantly more compact than the raw output of our algorithm. We do not claim that these simplifications constitute an exhaustive list of all possible simplifications. But Propositions 6.16 to 6.18 and Lemmas 6.19 and 6.20 cover all simplifications that are performed in our implementation of SPECIFICATION, and in the examples of Section 7.

7. Examples

We apply the method described in this article to find a combinatorial specification for three permutation classes. The first two are very small and easy examples, and are the examples of non substitution-closed permutation classes dealt with in [14] (Examples 4.3 and 4.4 of [14]). They use only a fraction of the machinery we have developed, but have been chosen to illustrate some of the differences between our method and the one of [14]. The third example is more involved, is meant to illustrate all of the auxiliary procedures defined, and shows better what our methodology brings.

7.1. Two examples from [14]
7.1.1. The class $C = \text{Av}(132)$

First, we note that $C$ is not substitution-closed (indeed, the excluded pattern 132 is not simple). Next, we remark that $C$ is a subclass of the separable permutations, that is to say it contains no simple permutations. (Equivalently, only nodes $\oplus$ and $\ominus$ appear in the decomposition trees.) Alternatively, this could be determined from steps 1 and 2 of the algorithmic chain presented in Section 3.2.

Because $C$ is not substitution-closed, a specification for $C$ cannot be simply derived as in Section 4, but can be obtained as the result of our algorithm SPECIFICATION. More precisely, the first step is to call EqnForRestriction($\hat{C}$, $\{132\}$, $\emptyset$) (which works exactly like EqnForClass on an input where the last part is $\emptyset$). It starts with Equation (1) of Proposition 4.2 (p.14), describing the substitution closure $\hat{C}$ of $C$, which can be written in our case as:

$$\hat{C} = 1 \uplus \oplus[\hat{C}^+, \hat{C}] \uplus \ominus[\hat{C}^-, \hat{C}]$$

Then, we propagate into each term of this equation the constraint of avoiding the pattern 132, using the embeddings of 132 into $\oplus$ and $\ominus$. There is only one non-trivial embedding in this case, corresponding to $132 = \oplus[1, 21]$. This results in the equation

$$C = \hat{C}\langle 132 \rangle = 1 \uplus \oplus[\hat{C}^+(132), \hat{C}(21)] \uplus \ominus[\hat{C}^-(132), \hat{C}(132)]$$

Because this equation is already non-ambiguous, calling the disambiguation procedure Disambiguate does not modify it (except for changing the union symbols $\uplus$ into disjoint union symbols $\uplus$). This is the first equation of our specification for $C$.

This equation contains right-only restrictions, like $\hat{C}^+(132)$, $\hat{C}(21)$, ... To complete the specification, SPECIFICATION next computes an equation for each of them, in the same fashion as above, that is to say running EqnForRestriction followed by Disambiguate. All right-only restrictions that appear are processed iteratively in the same way by SPECIFICATION, until none is left. Termination is guaranteed since the number of possible restrictions considered is finite (see Section 6.4).
The final result of calling Specification on \( C = Av(132) \) is the following specification:

\[
C = \hat{C}(132) = 1 \oplus \hat{C}^+(132), \hat{C}^-(132)] \oplus \hat{C}^+(132) \oplus [\hat{C}^-(132), \hat{C}(132)]
\]

\[
\hat{C}^+(132) = 1 \oplus \hat{C}^+(21)
\]

\[
\hat{C}^-(132) = 1 \oplus \hat{C}^+(132), \hat{C}(21)
\]

\[
\hat{C}(21) = 1
\]

An important remark is that this is exactly the result produced by our general method, without applying any ad hoc argument about this specific \( C \). Note that this specification is actually also the result of applying AmbiguousSystem to \( C \), which happens to be a non-ambiguous system already in this case.

We now turn to the specifications for \( C = Av(132) \) given in [14]. A specification for \( C \) may be derived following the proofs of [14]. As reminded in Section 3.1, the proof of their Lemma 2.1 shows that the set \( P_C = \{ Av(12) \}, Av(132), \mathcal{I}_0, \mathcal{I}_0 \} \) is query-complete, where \( \mathcal{I}_0 \) (resp. \( \mathcal{I}_0 \)) is the set of \( \ominus \)-indecomposable (resp. \( \ominus \)-indecomposable) permutations. Starting from this query-complete set, the proof of their Theorem 1.1 shows that there exists a specification for \( C \) having \( 1 + 2^4 = 17 \) equations. To obtain this specification, we would need to compute the sets \( E_{X, \sigma} \) defined in Section 3.1, which is not addressed in general in [14].

This specification is however not the one presented in Example 4.3 of [14]. Using ad hoc arguments and constructions, this example shows that the set \( P = \{ Av(21), Av(132), \mathcal{I}_0, \mathcal{I}_0 \} \) is query-complete. This allows to derive a specification for \( C \) with significantly fewer equations. Namely, removing empty terms like \( C_{I_0, I_0} \) or \( C_{Av(21), I_0} \) this specification has the same number of equations as ours; it is

\[
C = C_{Av(21), I_0, I_0} \oplus C_{Av(21), I_0} \oplus C_{I_0} \oplus C_{I_0, I_0} = 1
\]

\[
C_{Av(21), I_0, I_0} = \oplus \{ C_{Av(21), I_0, I_0} \}
\]

\[
C_{Av(21), I_0} = \oplus \{ C_{Av(21), I_0, I_0} \} \oplus C_{Av(21), I_0} \oplus C_{I_0} \oplus C_{I_0, I_0}
\]

\[
C_{I_0} = \oplus \{ C_{I_0}, C_{Av(21), I_0} \}
\]

In the system above, for \( X \subset P' = \{ Av(21), \mathcal{I}_0, \mathcal{I}_0 \} \), the set \( C_X \) represents permutations of \( C \) satisfying all properties in \( X \) and none in \( P' \setminus X \).

Finally, note that in both our construction and the specification given by [14, Example 4.3], the constraint of avoiding the pattern 21 appears because of the embedding of 132 into \( \ominus \) (corresponding to \( 132 = \ominus [1, 21] \)), which is the only non-trivial embedding in this example. However, our specification and the one of [14] are not the same. For example, [14] has an equation for \( C_{I_0} = C^-(21) \) whereas we do not have an equation for this set, but for \( \hat{C}^-(132) = C^- \). Note that beyond the difference of notation, the equations of [14] and ours have a different form. For example, our equation for \( \hat{C}^+(132) = C^+ \) correspond to the equation of [14] for \( C_{I_0} = C^+(21) = C^+ \setminus \{1\} \), but since we look at a coarser level, we express this later set as \( \ominus [\hat{C}^-(132), \hat{C}(132)] \) while it is expressed in [14] as \( \ominus [C_{Av(21), I_0, I_0} \oplus C_{Av(21), I_0} \oplus C_{I_0}, C_{Av(21), I_0} \oplus C_{Av(21), I_0} \oplus C_{I_0} \oplus C_{I_0}] \).
7.1.2. The class \( C = \text{Av}(2413, 3142, 2143) \)

This second example is another non substitution-closed subclass of the set of separable permutations. A specification for \( C \) can again be obtained using our algorithm Specification. The starting point is Equation (1) of Proposition 4.2 (p.14), in which the 2143-avoidance constraint has been propagated by EqnForRestriction (or equivalently EqnForClass in this simple case). This equation is

\[
C = \hat{C}(2143) = 1 \cup \oplus [\hat{C}^+ (2143), \hat{C}(21)] \cup \oplus [\hat{C}^+ (21), \hat{C}(2143)] \cup \oplus [\hat{C}^- (2143), \hat{C}(2143)],
\]

which is ambiguous. Indeed, the two terms \( \oplus [\hat{C}^+ (21), \hat{C}(2143)] \) and \( \oplus [\hat{C}^+ (2143), \hat{C}(21)] \) have a non-empty intersection (for instance, 12 belongs to both terms). The reason for this ambiguity is the embedding corresponding to 2143 = [21], which has two components different from \( \varepsilon \) and 1. (This is the only non-trivial embedding for the class \( C \) considered.)

Calling the procedure Disambiguate on the above equation (and performing the simplifications of Section 6.6), Specification obtains the first equation of a specification for \( \hat{C} \):

\[
C = \hat{C}(2143) = 1 \cup [\hat{C}^+ (2143)(21), \hat{C}(21)] \cup [\hat{C}^+ (21), \hat{C}(2143)(21)] \cup [\hat{C}^+ (21), \hat{C}(21)]
\]

\[
\cup [\hat{C}^+ (2143), \hat{C}(2143)].
\]

Note that restrictions with pattern containment constraints, like \( \hat{C}^+ (2143)(21) \), have appeared when running Disambiguate. The full specification is then obtained calling iteratively the procedures EqnForRestriction and Disambiguate (with simplifications) on all right-only restrictions. The final result produced by Specification for \( C = \text{Av}(2413, 3142, 2143) \) is then

\[
C = \hat{C}(2143) = 1 \cup [\hat{C}^+ (2143)(21), \hat{C}(21)] \cup [\hat{C}^+ (21), \hat{C}(2143)(21)] \cup [\hat{C}^+ (21), \hat{C}(21)]
\]

\[
\cup [\hat{C}^+ (2143), \hat{C}(2143)].
\]

\[
\hat{C}^+ (2143)(21) = \odot [\hat{C}^− (2143), \hat{C}(2143)]
\]

\[
\hat{C}(21) = 1 \cup [\hat{C}^+ (21), \hat{C}(21)]
\]

\[
\hat{C}^+ (21) = 1
\]

\[
\hat{C}(2143)(21) = \odot [\hat{C}^+ (2143)(21), \hat{C}(21)] \cup [\hat{C}^+ (21), \hat{C}(2143)(21)] \cup [\hat{C}^− (2143), \hat{C}(2143)]
\]

\[
\hat{C}^− (2143) = 1 \cup [\hat{C}^+ (21), \hat{C}(21)] \cup [\hat{C}^+ (2143)(21), \hat{C}(21)] \cup [\hat{C}^+ (21), \hat{C}(2143)(21)].
\]

This specification can again be compared with those obtained from [14]. The proofs therein show that the set \( P_C = \{ \text{Av}(12), \text{Av}(21), \text{Av}(132), \text{Av}(213), \text{Av}(231), \text{Av}(312), \text{Av}(2143), \text{Av}(3142), \text{Av}(2143), \text{I}_\oplus, \text{I}_\ominus \} \) is query-complete, so that there exists a specification for \( C \) having \( 1 + 2^8 = 257 \) equations.

Example 4.4 of [14] however provides a specification with fewer equations. It is derived from the query-complete set \( P = \{ \text{Av}(21), \text{Av}(2143), \text{I}_\oplus, \text{I}_\ominus \} \). There is however no hint of how this set was found, and especially not of a general method which would have resulted in this query-complete set. The specification for \( C \) that follows from this set through the proof of [14, Theorem 1.1] would then have \( 1 + 2^3 = 9 \) equations. Getting rid of empty terms, the specification for \( C \) obtained from this set which is given in [14,
Example 4.4] is:

\[ C = C_{\text{Av}(21)},\hat{I} \oplus C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},\hat{I} \oplus C_{\text{Av}(21)},I \]

\[ C_{\text{Av}(21)},I = 1 \]

\[ C_{\text{Av}(21)},\hat{I} = \hat{\bigoplus} \left[ C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},\hat{I} \right] \]

\[ C_{\text{Av}(21)},I = \bigoplus \left[ C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},\hat{I} \right] \]

\[ C_{\text{Av}(21)},\hat{I} = \bigoplus \left[ C_{\text{Av}(21)},I \oplus C_{\text{Av}(21)},\hat{I} \right] \]

In the system above, for \( X \subset P \)

\[ B \]

pattern avoidance constraints imposed by the avoidance of \( 2341 \)

are the only two non-simple patterns in \( B \)

\[ \hat{B} \]

From this description

we derive its generating function and furthermore generate at random large permutations

with this third example, we apply the method described in this article in a more complicated case. Namely, we derive a combinatorial specification for the permutation class \( C = \text{Av}(B) \) where \( B = \{1243, 2341, 2413, 41352, 531642\} \). From this description

we derive its generating function and furthermore generate at random large permutations

of the class. We follow the different steps described in the diagram of Figure 2 (p.11).

First, note that \( \text{Av}(B) \)

is not substitution-closed as 1243 and 2341 are non-simple permutations and belong to the basis of the class. Then we test whether the class contains a finite number of simple permutations, and if it is the case, we compute the set \( S_C \)

of simple permutations in \( C \). In our case, there is only one simple permutation in \( C \): \( S_C = \{3142\} \).

Now we have all we need to run our algorithm Specification, or our algorithm AmbiguousSystem if we want to compute the generating function using inclusion-exclusion. Their first step is to compute an equation for \( C \).

7.2. The class \( C = \text{Av}(1243, 2341, 2413, 41352, 531642) \)

An equation for the substitution-closure \( \hat{C} \) of \( C \). An equation for the substitution closure \( \hat{C} \) of \( C \) immediately follows from \( S_C \). Specifically, Equation (1) of Proposition 4.2 (p.14) is in our case:

\[ C = \hat{1} \oplus \hat{\bigoplus} \left[ \hat{C}^+, \hat{C} \right] \oplus \hat{\bigoplus} \left[ \hat{C}^-, \hat{C} \right] \oplus 3142[\hat{C}, \hat{C}, \hat{C}] \]

\[ (17) \]

From \( \hat{C} \) to \( C = \text{Av}(B) \). Since \( C = \hat{C}(B^*) \) where \( B^* \) denotes the non-simple permutations of \( B \), the algorithm AmbiguousSystem calls EqnForClass(\( \hat{C}, B^* \)). Similarly, the algorithm Specification calls EqnForRestriction(\( \hat{C}, B^*, \emptyset \)), which works exactly like EqnForClass(\( \hat{C}, B^* \)). Starting from Equation (17), it consists in adding the non-simple pattern avoidance constraints imposed by the avoidance of \( B \). In our case, 1243 and 2341 are the only two non-simple patterns in \( B \), and we compute an equation for \( C = \hat{C}(1243, 2341) \) by adding these two constraints one after the other.

\[ \text{The reader interested in more details about this example can find them in [26, Section 3.5].} \]
To compute $\hat{C}(1243)$, we propagate the constraint of avoiding 1243 into each term of Equation (17) using the embeddings of $\gamma = 1243$ into $\oplus$, $\ominus$, and $3142$. This gives the following equation:

$$\hat{C}(1243) = 1 \cup \oplus[\hat{C}^+(12),\hat{C}(132)] \cup \oplus[\hat{C}^+(1243),\hat{C}(21)] \cup \ominus[\hat{C}^-(1243),\hat{C}(1243)] \cup 3142[\hat{C}(1243),\hat{C}(12),\hat{C}(132),\hat{C}(132)].$$

As explained in Section 6.6, to obtain the above equation, two types of simplifications have been performed. First, when a union contains two terms such that one is strictly included in the other, then the smaller term has been removed (see Proposition 6.16). Second, when two excluded patterns are such that the avoidance of one implies the avoidance of the other, then the larger one has been removed (see Proposition 6.18). For instance we have simplified $\hat{C}(1243,12)$ into $\hat{C}(12)$.

Next we add the constraint of avoiding 2341 in the above equation as $\hat{C}(1243,2341) = \hat{C}(1243,2341)$. We propagate the constraint 2341 into the 5 different terms that appear in Equation (18), and perform simplifications as described above. The output of $\text{EqnForClass}(\hat{C}, B^+)$ or $\text{EqnForRestriction}(\hat{C}, B^+, \emptyset)$ is then:

$$\hat{C}(1243, 2341) = 1 \cup \oplus[\hat{C}^+(1243, 2341),\hat{C}(21)] \cup \oplus[\hat{C}^+(12),\hat{C}(132, 2341)] \cup \ominus[\hat{C}^-(1243, 2341),\hat{C}(12),\hat{C}(132),\hat{C}(132)].$$

If we run $\text{AmbiguousSystem}$, this is exactly the first equation of the system. This equation is ambiguous, since the intersection of $t_1 = \oplus[\hat{C}^+(12),\hat{C}(132, 2341)]$ and $t_2 = \oplus[\hat{C}^+(1243, 2341),\hat{C}(21)]$ is $\oplus[\hat{C}^+(12),\hat{C}(21)]$ which is not empty. $\text{Disambiguation}$. If we run $\text{Specification}$, the equation is disambiguated using the procedure $\text{Disambiguate}$. We can write $(t_1 \cup t_2)$ as $(t_1 \cap t_2) \oplus (t_1 \cap t_2) \oplus (t_1 \cap t_2)$. The computation of the complement terms $t_1$ and $t_2$ (with Equation (15) p.31) increases the number of terms in the disjoint union that describes $t_1 \cup t_2$, producing in this case a disjoint union of 15 terms. After eliminating terms that are empty or strictly included in another one, Equation (19) is replaced by:

$$\hat{C}(1243, 2341) = 1 \cup \oplus[\hat{C}^+(1243, 2341)(12),\hat{C}(21)] \cup \oplus[\hat{C}^+(12),\hat{C}(132, 2341)(21)] \cup \oplus[\hat{C}^+(12),\hat{C}(21)] \cup \ominus[\hat{C}^-(123),\hat{C}(1243, 2341)] \cup 3142[\hat{C}(12),\hat{C}(12),\hat{C}(132, 2341)].$$

This is the final version of the first equation of the specification.

### 7.2.2. The whole system

$\text{AmbiguousSystem}$. To obtain a complete system, $\text{AmbiguousSystem}$ starts with Equation (19) computed above. Using $\text{EqnForClass}$, it iterates the process of computing equations for new classes $\hat{C}^E$ appearing on the right side of this equation, and it does it subsequently for any class appearing in one of these new equations. As already mentioned, this process terminates since there is a finite number of subsets $E$ of patterns of permutations of $B$. 

40
The result of AmbiguousSystem on $C$ is the following:

$$
\hat{C}(124, 2341) = 1 \cup \oplus [\hat{C}^+(1243, 2341), \hat{C}(21)] \cup \oplus [\hat{C}^+(12), \hat{C}(132, 2341)]
$$

$$
\hat{C}(21) = 1 \cup \oplus [\hat{C}^+(21), \hat{C}(21)]
$$

$$
\hat{C}^+(124, 2341) = 1 \cup \oplus [\hat{C}^+(123), \hat{C}(1243, 2341)] \cup 3142[\hat{C}(12), \hat{C}(12), \hat{C}(132, 2341)]
$$

$$
\hat{C}^+(123) = 1 \cup \oplus [\hat{C}^+(12), \hat{C}(12)] \cup 3142[\hat{C}(12), \hat{C}(12), \hat{C}(12)]
$$

$$
\hat{C}(21) = 1 \cup \oplus [\hat{C}^+(21), \hat{C}(21)]
$$

$$
\hat{C}^+(132, 2341) = 1 \cup \oplus [\hat{C}^+(132, 2341), \hat{C}(21)] \cup \oplus [\hat{C}^+(132, 123), \hat{C}(132, 2341)]
$$

$$
\hat{C}^+(132, 123) = 1 \cup \oplus [\hat{C}^+(12), \hat{C}(21, 12)]
$$

$$
\hat{C}^+(21, 12) = 1
$$

As noticed earlier, the first equation of the system is indeed ambiguous. This system can be used to compute the generating function of $\hat{C}$ using inclusion-exclusion. Since we want not only to compute the generating function of $\hat{C}$, but also to generate at random uniform permutations of $\hat{C}$, we rather compute a specification for $\hat{C}$.

**Specification.** The unambiguous equation for $\hat{C}$ computed by Specification is Equation (20). As noticed in Section 6.3, right-only terms, possibly involving pattern containment constraints like $\hat{C}(132, 2341)(21)$, appear in this equation. These terms are not defined by our system, so we have to compute an equation for each of them (and iteratively so for right-only terms appearing in these new equations). This is done by iterating EqnForRestriction. This procedure works in a fashion similar to EqnForClass, except that, in addition, it propagates pattern containment constraints in the subtrees. For the term $\hat{C}(132, 2341)(21)$, we describe how Algorithm EqnForRestriction computes an additional equation. We do not give details for the other right-only terms.

First, starting from Equation (17), we add the pattern avoidance constraints, namely 132 and 2341, and compute an equation for $\hat{C}(132, 2341)$ like before. We obtain the following equation:

$$
\hat{C}(132, 2341) = 1 \cup \oplus [\hat{C}^+(132, 2341), \hat{C}(21)] \cup \oplus [\hat{C}^+(132, 123), \hat{C}(132, 2341)].
$$

Then, we add the pattern containment constraints (here 21) to this equation, considering embeddings of 21 in $\oplus$ and $\ominus$. The equation obtained after simplification is

$$
\hat{C}(132, 2341)(21) = \ominus [\hat{C}^+(132, 2341)(21), \hat{C}(21)] \cup \ominus [\hat{C}^+(132, 123), \hat{C}(132, 2341)].
$$

This equation is already unambiguous, so this is its final version.

The process is iterated until each term that appear on the right hand side of an equation of the system is defined by an unambiguous equation, which finally leads to the following specification:
\[ \hat{C}(1243, 2341) = 1 \oplus \hat{C}^+(12), \hat{C}(132, 2341)(21) \oplus \hat{C}^+(1243, 2341)(12), \hat{C}(21) \oplus \hat{C}^+(12), \hat{C}(21) \]
\[ \hat{C}^+(12) = 1 \oplus \hat{C}^-(12), \hat{C}(12) \]
\[ \hat{C}(132, 2341)(21) = \oplus \hat{C}^+(132, 2341)(21), \hat{C}(21) \oplus \oplus \hat{C}^-(132, 123), \hat{C}(132, 2341) \]
\[ \hat{C}^+(1243, 2341)(12) = \oplus \hat{C}^-(123)(12), \hat{C}(1243, 2341)(12) \oplus \oplus \hat{C}^-(12), \hat{C}(1243, 2341)(12) \]
\[ \hat{C}^+(21) = 1 \oplus \hat{C}^+(21), \hat{C}(21) \]
\[ \hat{C}^-(123) = 1 \oplus \hat{C}^+(12), \hat{C}(12) \oplus \hat{3142[\hat{C}(12), \hat{C}(12), \hat{C}(12)] \]
\[ \hat{C}(132, 2341) = 1 \oplus \oplus \hat{C}^+(132, 2341), \hat{C}(21) \oplus \hat{C}^-(132, 123), \hat{C}(132, 2341) \]
\[ \hat{C}^-(12) = 1 \]
\[ \hat{C}^+(132, 2341)(21) = \oplus \hat{C}^-(132, 123), \hat{C}(132, 2341) \]
\[ \hat{C}^+(132, 123) = 1 \oplus \hat{C}^+(12), \hat{C}(21), \hat{C}(12) \]
\[ \hat{C}^-(123)(12) = \oplus \hat{C}^+(12), \hat{C}(12) \oplus \hat{3142[\hat{C}(12), \hat{C}(12), \hat{C}(12)] \]
\[ \hat{C}(1243, 2341)(12) = \oplus \hat{C}^+(12), \hat{C}(1243, 2341)(21) \oplus \hat{3142[\hat{C}(12), \hat{C}(12), \hat{C}(12), \hat{C}(132, 2341)] \]
\[ \hat{C}^+(21) = 1 \]
\[ \hat{C}^+(132, 2341) = 1 \oplus \oplus \hat{C}^-(132, 123), \hat{C}(132, 2341) \]
\[ \hat{C}(21, 21) = 1. \]

The above specification contains 16 equations. This should be compared to the number of equations in the specification that could be derived using the general method described in the proofs of [14] and reviewed in Section 3.1 (and which would require to compute the sets \( E_x \) defined in Section 3.1). The number of equations following this method would be \( a \text{ priori} \ 1 + 2^k \) where \( k \) is the number of proper patterns of permutations in \( B \) (in this case, \( k = 27 \), giving 536 870 913 equations), although we can expect many empty terms and hence fewer equations.

7.2.3. Byproducts of the combinatorial specification

The translation of the above specification into a system of equations for the generating function of \( \hat{C} \) produces in this specific case a system that can be solved, giving access to a closed form for the generating function of \( \hat{C} \):

\[ C(z) = \frac{(z^6 - 7z^5 + 20z^4 - 28z^3 + 20z^2 - 7z + 1)z}{1 - 9z + 32z^2 - 59z^3 + 62z^4 - 37z^5 + 13z^6 - 2z^7}. \]

We can also translate this specification into (Boltzmann or recursive) uniform random samplers of permutations in \( \hat{C} \). We have used the recursive sampler\(^{11}\) so obtained to uniformly generate permutations in \( \hat{C} \). This is how Figure 3 (p.12) was obtained, as well as Figure 4 (p.13) for other examples of classes \( \hat{C} \).

\(^{11}\)In this case (and in the one of Figure 4), we chose a recursive sampler over a Boltzmann one because it is more suitable when you want to sample many random permutations of the same medium size.
Acknowledgments

We are grateful to the anonymous referees and to Mireille Bousquet-Mélou, whose comments have helped us make the relation between our work and [14] clearer.

Funding: This work was completed with the support of the ANR [project MAGNUM number 2010_BLAN_0204].


