Combinatorial specification of permutation classes[†]

Frédérique Bassino[1] Mathilde Bouvel[2] Adeline Pierrot[3]
Carine Pivoteau[4] Dominique Rossin[5]

[1] *Université Paris 13, LIPN (CNRS UMR 7030), Villetaneuse, France.*
[2] *Université de Bordeaux, LaBRI (CNRS UMR 5800), Talence, France.*
[3] *Université Paris Diderot, LIAFA (CNRS UMR 7089), Paris, France.*
[4] *Université Paris-Est, LIGM (CNRS UMR 8049), Marne-la-Vallée, France.*
[5] *École Polytechnique, LIX (CNRS UMR 7161), Palaiseau, France.*

Abstract. This article presents a methodology that automatically derives a combinatorial specification for the permutation class $\mathcal{C} = Av(B)$, given its basis B of excluded patterns and the set of simple permutations in $\mathcal{C}$, 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 $\mathcal{C}$, this system being always positive and algebraic. It also yields a uniform random sampler of permutations in $\mathcal{C}$. The method presented is fully algorithmic.

Résumé. Cet article présente une méthodologie qui calcule automatiquement une spécification combinatoire pour la classe de permutations $\mathcal{C} = Av(B)$, étant donnés une base B de motifs interdits et l'ensemble des permutations simples de $\mathcal{C}$, lorsque ces deux ensembles sont finis. Ce résultat est obtenu en considérant à la fois des contraintes de motifs interdits et de motifs obligatoires dans les permutations. La spécification obtenue donne un système d'équations satisfait par la série génératrice de la classe $\mathcal{C}$, système qui est toujours positif et algébrique. Elle fournit aussi un générateur aléatoire uniforme de permutations dans $\mathcal{C}$. La méthode présentée est complètement algorithmique.

Keywords: permutation classes, excluded patterns, substitution decomposition, simple permutations, generating functions, combinatorial specification, random generation

1 Introduction

Initiated by Knuth (1973) almost forty years ago, the study of permutation classes has since received a lot of attention, mostly with respect to enumerative questions (see Bousquet-Mélou (2002); Elizalde (2004); Kitaev and Mansour (2003) and their references among many others). Most articles are focused on a given class $\mathcal{C} = Av(B)$ where the basis B of excluded patterns characterizing $\mathcal{C}$ is finite, explicit, and in most cases contains only patterns of size 3 or 4. Recently, some results of a rather different nature have been obtained, and have in common that they describe general properties of permutation classes – see Albert and Atkinson (2005); Albert et al. (2005); Bassino et al. (2010, 2011); Brignall et al. (2008a,b); Vatter (2008) for example. Our work falls into this new line of research.

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

Our goal in this article is to provide a general algorithmic method to obtain a combinatorial specification for any permutation class $\mathcal{C}$ from its basis B and the set $\mathcal{S}_\mathcal{C}$ of simple permutations in $\mathcal{C}$, and assuming these two sets are finite. Notice that by previous works to be detailed in Section 3, it is enough to know the finite basis B of the class to decide whether the set $\mathcal{S}_\mathcal{C}$ is finite and (in the affirmative) to compute $\mathcal{S}_\mathcal{C}$.

By *combinatorial specification* of a class (see Flajolet and Sedgewick (2009)), we mean an unambiguous system of combinatorial equations that describe recursively the permutations of $\mathcal{C}$ using only combinatorial constructors (disjoint union, cartesian product, sequence, ...) and permutations of size 1. Notice the major difference with the results of Albert and Atkinson (2005): our specifications are unambiguous, whereas Albert and Atkinson (2005) obtain combinatorial systems of equations characterizing permutations classes that are ambiguous in general.

We believe that our purpose of obtaining algorithmically combinatorial specifications of permutation classes is of interest *per se* but also because it then allows to obtain by routine algorithms a system of equations satisfied by the generating function of $\mathcal{C}$ and a Boltzmann uniform random sampler of permutations in $\mathcal{C}$, using the methods of Flajolet and Sedgewick (2009) and Duchon et al. (2004) respectively.

The paper is organized as follows. Section 2 proceeds with some background on permutation classes, simple permutations and substitution decomposition, and Section 3 sets the algorithmic context of our study. Section 4 then explains how to obtain a system of combinatorial equations describing $\mathcal{C}$ from the set of simple permutations in $\mathcal{C}$, that we assume to be finite. The system so obtained may be ambiguous and Section 5 describes a disambiguation algorithm to obtain a combinatorial specification for $\mathcal{C}$. The most important idea of this disambiguation procedure is to transform ambiguous unions into disjoint unions of terms that involve both pattern avoidance and pattern containment constraints. This somehow allows to interpret on the combinatorial objects themselves the result of applying the inclusion-exclusion on their generating functions. Finally, Section 6 concludes the whole algorithmic process by explaining how this specification can be plugged into the general methodologies of Flajolet and Sedgewick (2009) and Duchon et al. (2004) to obtain a system of equations satisfied by the generating function of $\mathcal{C}$ and a Boltzmann uniform random sampler of permutations in $\mathcal{C}$. We also give a number of perspectives opened by our algorithm.

2 Permutation classes and simple permutations

2.1 Permutation patterns and permutation classes

A permutation $\sigma = \sigma_1\sigma_2\dots\sigma_n$ of size $|\sigma| = n$ is a bijective map from $\{1,\dots,n\}$ to itself, each σ_i denoting the image of i under σ. A permutation $\pi = \pi_1\pi_2\dots\pi_k$ is a *pattern* of a permutation $\sigma = \sigma_1\sigma_2\dots\sigma_n$ (denoted $\pi \preceq \sigma$) if and only if $k \leq n$ and there exist integers $1 \leq i_1 < i_2 < \dots < i_k \leq n$ such that $\sigma_{i_1}\dots\sigma_{i_k}$ is order-isomorphic to π, *i.e.* such that $\sigma_{i_\ell} < \sigma_{i_m}$ whenever $\pi_\ell < \pi_m$. A permutation σ that does not contain π as a pattern is said to *avoid* π. For example the permutation $\sigma = 316452$ contains $\pi = 2431$ as a pattern, whose occurrences are 3642 and 3652. But σ 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 a *permutation class* $\mathcal{C}$ is a downset under this order: for any $\sigma \in \mathcal{C}$, if $\pi \preceq \sigma$, then we also have $\pi \in \mathcal{C}$. For every set B, the set $Av(B)$ of permutations avoiding any pattern of B is a class. Furthermore every class $\mathcal{C}$ can be rewritten as $\mathcal{C} = Av(B)$ for a unique antichain B (*i.e.,* a unique set of pairwise incomparable elements) called the *basis* of $\mathcal{C}$. The basis of a class $\mathcal{C}$ may be finite or infinite; it is described as the set of permutations that do not belong to $\mathcal{C}$ and that are minimal in the sense of $\preceq$ for this criterion.

In the following, we only consider classes whose basis B is given explicitly, and is finite. This does not cover the whole range of permutation classes, but it is a reasonable assumption when dealing with *algorithms* on permutation classes, that take a finite description of a permutation class as input. Moreover, as proved by Albert and Atkinson (2005), it is necessary that B is finite as soon as the set $\mathcal{S}_\mathcal{C}$ of simple permutations in $\mathcal{C} = Av(B)$ is finite. Consequently the assumption of the finiteness of B is not a restriction when working on permutation classes such that $\mathcal{S}_\mathcal{C}$ is finite, which is the context of our study.

2.2 Simple permutations and substitution decomposition of permutations

An *interval* (or *block*) of a permutation σ of size n is a subset $\{i, \ldots, (i + \ell - 1)\}$ of consecutive integers of $\{1, \ldots, n\}$ whose images by σ also form an interval of $\{1, \ldots, n\}$. The integer ℓ is called the *size* of the interval. A permutation σ is *simple* when it is of size at least 4 and it contains no interval, except the trivial ones: those of size 1 (the singletons) or of size n (σ itself). The permutations 1, 12 and 21 also have only trivial intervals, nevertheless they are *not* considered to be simple here. Moreover no permutation of size 3 has only trivial intervals. For a detailed study of simple permutations, in particular from an enumerative point of view, we refer the reader to Albert and Atkinson (2005); Albert et al. (2003); Brignall (2010).

Let σ 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 σ to be the permutation of size $p_1 + \ldots + p_n$ 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 an interval, are ordered in a sequence order-isomorphic to π^i, and S^i consists of integers smaller than S^j if and only if $\sigma_i < \sigma_j$. For instance, the substitution $1\,3\,2[2\,1, 1\,3\,2, 1]$ gives the permutation 2 1 4 6 5 3. We say that a permutation π is 12-*indecomposable* (resp. 21-*indecomposable*) if it cannot be written as $12[\pi^1, \pi^2]$ (resp. $21[\pi^1, \pi^2]$), for any permutations π^1 and π^2.

Simple permutations allow to describe all permutations through their *substitution decomposition*.

Theorem 2.1 (Albert and Atkinson (2005)) *Every permutation π of size n with $n \geq 2$ can be uniquely decomposed as follows,* 12 *(resp.* 21*, σ) being called the* root *of π:*

- $12[\pi^1, \pi^2]$*, with π^1* 12*-indecomposable,*
- $21[\pi^1, \pi^2]$*, with π^1* 21*-indecomposable,*
- $\sigma[\pi^1, \pi^2, \ldots, \pi^k]$*, with σ a simple permutation of size k.*

To account for the first two items of Theorem 2.1 in later discussions, we furthermore introduce the following notations: For any set $\mathcal{C}$ of permutations, $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) denotes the set of permutations of $\mathcal{C}$ that are 12-indecomposable (resp. 21-indecomposable). Notice that even when $\mathcal{C}$ is a permutation class, this is not the case for $\mathcal{C}^+$ and $\mathcal{C}^-$ in general.

Theorem 2.1 provides the first step in the decomposition of a permutation π. To obtain its full decomposition, we can recursively decompose the permutations π^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 π. Each internal node of the tree is labeled by $12, 21$ or by a simple permutation and the leaves represent permutation 1. Notice that in decomposition trees, the left child of a node labeled 12 (resp. 21) is never labeled by 12 (resp. 21), since π^1 is 12-indecomposable (resp. 21-indecomposable) in the first (resp. second) item of Theorem 2.1.

Example 2.2 *The permutation $\pi = 8\ 9\ 5\ 11\ 7\ 6\ 10\ 17\ 2\ 1\ 3\ 4\ 14\ 16\ 13\ 15\ 12$ is recursively decomposed as* $\pi = 2413[4517326, 1, 2134, 35241] = 2413[31524[12[1, 1], 1, 1, 21[1, 1], 1]], 1, 12[21[1, 1], 12[1, 1]],$ $21[2413[1, 1, 1, 1], 1]]$ *and its decomposition tree is given in Figure 1.*

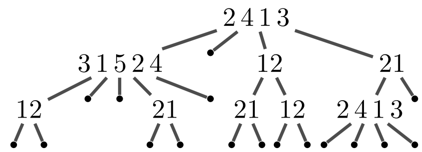

Fig. 1: Decomposition tree of π (from Ex. 2.2).

The *substitution closure* $\hat{\mathcal{C}}$ of a permutation class[(i)] $\mathcal{C}$ is defined as the set of permutations whose decomposition trees have internal nodes labeled by either $12, 21$ or a simple permutation of $\mathcal{C}$. Notice that $\mathcal{C}$ and $\hat{\mathcal{C}}$ therefore contain the same simple permutations. Obviously, for any class $\mathcal{C}$, we have $\mathcal{C} \subseteq \hat{\mathcal{C}}$. When the equality holds, the class $\mathcal{C}$ is said to be *substitution-closed* (or sometimes *wreath-closed*). But this is not always the case, and the simplest example is given by $\mathcal{C} = Av(213)$. This class contains no simple permutation hence its substitution closure is the class of separable permutations of Bose et al. (1998), *i.e.* of permutations whose decomposition trees have internal nodes labeled by 12 and 21. It is immediate to notice that $213 \in \hat{\mathcal{C}}$ whereas of course $213 \notin \mathcal{C}$.

A characterization of substitution-closed classes useful for our purpose is given in Albert and Atkinson (2005): A class is substitution-closed if and only if its basis contains only simple permutations.

3 Algorithmic context of our work

Putting together the work reported in this article and recent algorithms from the litterature provides a full algorithmic chain starting with the finite basis B of a permutation class $\mathcal{C}$, and computing a specification for $\mathcal{C}$. The hope for such a very general algorithm is of course very tenuous, and the algorithm we describe below will compute its output only when some hypothesis are satisfied, which are also tested algorithmically. Figure 2 summarizes the main steps of the algorithm.

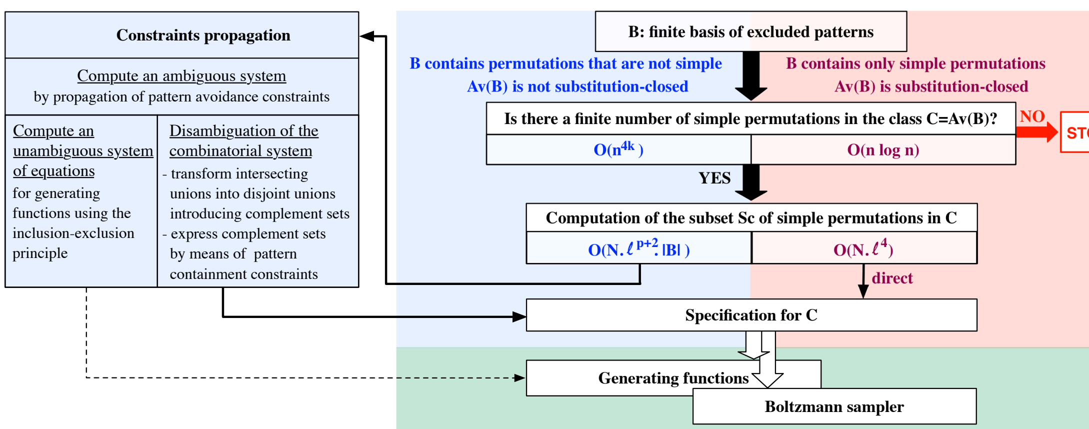

Fig. 2: Automatic process from the basis of a permutation class to generating function and Boltzmann sampler.

The algorithms performing the first two steps of the algorithmic process of Figure 2 are as follows.

First step : Finite number of simple permutations First, we check whether $\mathcal{C} = Av(B)$ contains only a finite number of simple permutations. This is achieved using algorithms of Bassino et al. (2010)

[(i)] that contains permutations 12 and 21. We will assume so in the rest of this article to avoid trivial cases.

when the class is substitution-closed and of Bassino et al. (2011) otherwise. The complexity of these algorithms are respectively $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^{4k})$, where $n = \sum_{\beta \in B} |\beta|$ and $k = |B|$.

Second step : Computing simple permutations The second step of the algorithm is the computation of the set of simple permutations $\mathcal{S}_\mathcal{C}$ contained in $\mathcal{C} = Av(B)$, when we know it is finite. Again, when $\mathcal{C}$ is substitution-closed, $\mathcal{S}_\mathcal{C}$ can be computed by an algorithm that is more efficient than in the general case. The two algorithms are described in Pierrot and Rossin (2012), and their complexity depends on the output: $\mathcal{O}(N \cdot \ell^{p+2} \cdot |B|)$ in general and $\mathcal{O}(N \cdot \ell^4)$ for substitution-closed classes, with $N = |\mathcal{S}_\mathcal{C}|$, $p = \max\{|\beta| : \beta \in B\}$ and $\ell = \max\{|\pi| : \pi \in \mathcal{S}_\mathcal{C}\}$.

Sections 4 and 5 will then explain how to derive a specification for $\mathcal{C}$ from $\mathcal{S}_\mathcal{C}$.

4 Ambiguous combinatorial system describing $\mathcal{C}$

We describe here an algorithm that takes as input the set $\mathcal{S}_\mathcal{C}$ of simple permutations in a class $\mathcal{C}$ and the basis B of $\mathcal{C}$, and that produces in output a (possibly ambiguous) system of combinatorial equations describing the permutations of $\mathcal{C}$ through their decomposition trees. The main ideas are those of Theorem 10 of Albert and Atkinson (2005), but unlike this work, we make the whole process fully algorithmic.

4.1 The simple case of substitution-closed classes

Recall that $\mathcal{C}$ is a substitution-closed permutation class when $\mathcal{C} = \hat{\mathcal{C}}$, or equivalently when the permutations in $\mathcal{C}$ are exactly the ones whose decomposition trees have internal nodes labeled by $12, 21$ or any simple permutation of $\mathcal{C}$. Then Theorem 2.1 directly yields the following system $\mathcal{E}_{\hat{\mathcal{C}}}$:

$$\hat{\mathcal{C}} = 1 \uplus 12[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}] \uplus 21[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}] \uplus \biguplus_{\pi \in \mathcal{S}_{\hat{\mathcal{C}}}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}] \quad (1)$$

$$\hat{\mathcal{C}}^+ = 1 \uplus 21[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}] \uplus \biguplus_{\pi \in \mathcal{S}_{\hat{\mathcal{C}}}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}] \quad (2)$$

$$\hat{\mathcal{C}}^- = 1 \uplus 12[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}] \uplus \biguplus_{\pi \in \mathcal{S}_{\hat{\mathcal{C}}}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]. \quad (3)$$

By uniqueness of substitution decomposition, unions are disjoint and so Equations (1) to (3) describe unambiguously the substitution closure $\hat{\mathcal{C}}$ of a permutation class $\mathcal{C}$. For a substitution-closed class (and the substitution closure of any class), this description gives a combinatorial specification. Hence, it provides an efficient way to compute the generating function of the class, and to generate uniformly at random a permutation of a given size in the class.

4.2 Adding constraints for classes that are not substitution-closed

When $\mathcal{C}$ is not substitution-closed, we compute a new system by adding constraints to the system obtained for $\hat{\mathcal{C}}$, as in Albert and Atkinson (2005). Denoting by $X\langle Y \rangle$ the set of permutations of X that avoid the patterns in Y, we have $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star \rangle$ where $B^\star$ is the subset of non-simple permutations of B. Noticing that $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$ (by definition of $\hat{\mathcal{C}}$), and since $\mathcal{C}^\varepsilon = \hat{\mathcal{C}}^\varepsilon\langle B^\star \rangle$ for $\varepsilon \in \{\ ,+,-\}$, Equations (1) to (3) give

$$\hat{\mathcal{C}}\langle B^\star \rangle = 1 \uplus 12[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}]\langle B^\star \rangle \uplus 21[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}]\langle B^\star \rangle \uplus \biguplus_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star \rangle \quad (4)$$

$$\hat{\mathcal{C}}^+\langle B^\star \rangle = 1 \uplus 21[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}]\langle B^\star \rangle \uplus \biguplus_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star \rangle \quad (5)$$

$$\hat{\mathcal{C}}^-\langle B^\star \rangle = 1 \uplus 12[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}]\langle B^\star \rangle \uplus \biguplus_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star \rangle, \quad (6)$$

all these unions being disjoint. This specification is not complete, since sets of the form $\pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star\rangle$ are not immediately described from $\hat{\mathcal{C}}\langle B^\star\rangle$. Theorem 10 of Albert and Atkinson (2005) explains how sets such as $\pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star\rangle$ can be expressed as union of smaller sets:

$$\pi[\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^\star\rangle = \bigcup_{i=1}^{k} \pi[\hat{\mathcal{C}}\langle E_{i,1}\rangle, \hat{\mathcal{C}}\langle E_{i,2}\rangle, \dots, \hat{\mathcal{C}}\langle E_{i,k}\rangle]$$

where $E_{i,j}$ are sets of permutations which are patterns of some permutations of $B^\star$. This introduces sets of the form $\hat{\mathcal{C}}\langle E_{i,j}\rangle$ on the right-hand side of an equation of the system that do not appear on the left-hand side of any equation. We will call such sets *right-only* sets. Taking $E_{i,j}$ instead of $B^\star$ in Equations (4) to (6), we can recursively compute these right-only sets by introducing new equations in the system. This process terminates since there exists only a finite number of sets of patterns of elements of $B^\star$ (as B is finite). Let us introduce some definitions to describe these sets $E_{i,j}$.

A *generalized substitution* $\sigma\{\pi^1,\pi^2,\dots,\pi^n\}$ is defined as a substitution (see p.783) with the particularity that any π^i may be the empty permutation (denoted by 0). Specifically $\sigma[\pi^1,\pi^2,\dots,\pi^n]$ necessarily contains σ whereas $\sigma\{\pi^1,\pi^2,\dots,\pi^n\}$ may avoid σ. For instance, $1\,3\,2\{2\,1,0,1\} = 2\,1\,3 \in Av(132)$.

An *embedding of* γ *in* $\pi = \pi_1\dots\pi_n$ is a map α from $\{1,\dots,n\}$ to the set of (possibly empty) blocks[(ii)] of γ such that:

- if blocks $\alpha(i)$ and $\alpha(j)$ are not empty, and $i<j$, then $\alpha(i)$ consists of smaller indices than $\alpha(j)$;
- as a word, $\alpha(1)\dots\alpha(n)$ is a factorization of the word $1\dots|\gamma|$ (which may include empty factors).
- denoting γ_I the pattern corresponding to $\gamma_{i_1}\dots\gamma_{i_\ell}$ for any block I of indices from i_1 to i_ℓ in increasing order, we have $\pi\{\gamma_{\alpha(1)},\dots,\gamma_{\alpha(n)}\} = \gamma$.

There are 11 embeddings of $\gamma = 5\,4\,6\,3\,1\,2$ into $\pi = 3\,1\,4\,2$, which correspond for instance to the generalized substitutions $\pi\{3241,12,0,0\}$, $\pi\{3241,0,0,12\}$ and $\pi\{0,0,3241,12\}$ for the same expression of γ as the substitution $21[3241,12]$, or $\pi\{3241,1,0,1\}$ which is the only one corresponding to $312[3241,1,1]$. Notice that this definition of embeddings conveys the same notion than in Albert and Atkinson (2005), but it is formally different and it will turn to be more adapted to the definition of the sets $E_{i,j}$.

Equations (4) to (6) can be viewed as Equations (1) to (3) "decorated" with pattern avoidance constraints. These constraints apply to every set $\pi[\hat{\mathcal{C}}_1,\dots,\hat{\mathcal{C}}_n]$ that appears in a disjoint union on the right-hand side of an equation. For each such set, the pattern avoidance constraints can be expressed by pushing constraints into the subtrees, using embeddings of excluded patterns in the root π. For instance, assume that $\gamma = 5\,4\,6\,3\,1\,2 \in B^\star$ and $\mathcal{S}_{\mathcal{C}} = \{3142\}$, and consider $3142[\hat{\mathcal{C}},\hat{\mathcal{C}},\hat{\mathcal{C}},\hat{\mathcal{C}}]\langle\gamma\rangle$. The embeddings of γ in 3142 indicates how pattern γ can be found in the subtrees in $3142[\hat{\mathcal{C}},\hat{\mathcal{C}},\hat{\mathcal{C}},\hat{\mathcal{C}}]$. As example the last embedding of the previous example tells that γ can spread over all the subtrees of 3142 except the third. In order to avoid this particular embedding of γ, it is enough to avoid one of the induced pattern γ_I on one of the subtrees. However, in order to ensure that γ is avoided, the constraints resulting from all the embeddings must be considered and merged. More precisely, consider a set $\pi[\mathcal{C}_1,\dots,\mathcal{C}_n]\langle\gamma\rangle$, π being a simple permutation. Let $\{\alpha_1,\dots,\alpha_\ell\}$ be the set of embeddings of γ in π, each α_i being associated to a generalized substitution $\gamma = \pi\{\gamma_{\alpha_i(1)},\dots,\gamma_{\alpha_i(n)}\}$ where $\gamma_{\alpha_i(k)}$ is embedded in π_k. Then the constraints are propagated according to the following equation:

$$\pi[\mathcal{C}_1,\dots,\mathcal{C}_n]\langle\gamma\rangle = \bigcup_{(k_1,\dots,k_\ell)\in K_\gamma^\pi} \pi[\mathcal{C}_1\langle E_{1,k_1\dots k_\ell}\rangle,\dots,\mathcal{C}_n\langle E_{n,k_1\dots k_\ell}\rangle] \tag{7}$$

where $K_\gamma^\pi = \{(k_1,\dots,k_\ell)\in[1..n]^\ell \mid \forall i,\ \gamma_{\alpha_i(k_i)} \neq 0\}$ and $E_{m,k_1\dots k_\ell} = \{\gamma_{\alpha_i(k_i)} \mid i\in[1..\ell] \text{ and } k_i = m\}$ is a set containing at least γ for $(k_1,\dots,k_\ell)\in K_\gamma^\pi$. In a tuple $(k_1,\dots,k_\ell)$ of K_γ^π, k_i indicates a

(ii) Recall that here blocks of a permutation are sets of *indices*.

subtree of π where the pattern avoidance constraint ($\gamma_{\alpha_i(k_i)}$ excluded) forbids any occurrence of γ that could result from the embedding α_i. The set $E_{m,k_1\dots k_\ell}$ represents the pattern avoidance constraints that have been pushed into the m-th subtree of π by embeddings α_i of γ in π where the block $\alpha_i(k_i)$ of γ is embedded into π_m.

Starting from a finite basis of patterns B, Algorithm 1 describes the whole process to compute an ambiguous system defining the class $\mathcal{C} = Av(B)$ knowing its set of simple permutations $\mathcal{S}_{\mathcal{C}}$. The propagation of the constraints expressed by Equation (7) is performed by the procedure ADDCONSTRAINTS. It is applied to every set of the form $\pi[\mathcal{C}_1, \dots, \mathcal{C}_n]\langle B'\rangle$ that appears in the equation defining some $\hat{\mathcal{C}}^\varepsilon\langle B'\rangle$ by the procedure COMPUTEEQN. Finally, Algorithm 1 computes an ambiguous system for a permutation class $Av(B)$ containing a finite number of simple permutations: it starts from Equations (4) to (6), and adds new equations to this system calling procedure COMPUTEEQN, until every $\pi[\mathcal{C}_1, \dots, \mathcal{C}_n]\langle B'\rangle$ is replaced by some $\pi[\mathcal{C}'_1, \dots, \mathcal{C}'_n]$ and until every $\mathcal{C}'_i = \hat{\mathcal{C}}^\varepsilon\langle B'_i\rangle$ is defined by an equation of the system. All the sets B' are sets of patterns of some permutations in B. Since there is only a finite number of patterns of elements of B, there is a finite number of possible B', and Algorithm 1 terminates.

Algorithm 1: AMBIGUOUSSYSTEM(B)

Data: B is a finite basis of patterns defining $\mathcal{C} = Av(B)$ such that $\mathcal{S}_{\mathcal{C}}$ is known and finite.
Result: A system of equations of the form $\mathcal{D} = \bigcup \pi[\mathcal{D}_1, \dots, \mathcal{D}_n]$ defining $\mathcal{C}$.

```
begin
    E ← COMPUTEEQN((Ĉ, B*)) ∪ COMPUTEEQN((Ĉ+, B*)) ∪ COMPUTEEQN((Ĉ−, B*))
    while there is a right-only Ĉ^ε⟨B'⟩ in some equation of E do
        E ← E ∪ COMPUTEEQN(Ĉ^ε, B')
end

/* Returns an equation defining Ĉ^ε⟨B'⟩ as a union of π[C_1, ..., C_n] */
/* B' is a set of permutations, Ĉ^ε is given by S_Ĉ and ε ∈ { , +, −} */
COMPUTEEQN (Ĉ^ε, B')
    E ← Equation (4) or (5) or (6) (depending on ε) written with B' instead of B*
    foreach t = π[C_1, ..., C_n]⟨B'⟩ that appears in E do
        t ← ADDCONSTRAINTS(π[C_1, ..., C_n], B')
    return E
end

/* Returns a rewriting of π[C_1 ... C_n]⟨E⟩ as a union ⋃ π[D_1, ... D_n] */
ADDCONSTRAINTS ((π[C_1 ... C_n], E))
    if E = ∅ then return π[C_1 ... C_n];
    else
        choose γ ∈ E and compute all the embeddings of γ in π
        compute K_γ^π and sets E_{m,k_1...k_ℓ} defined in Equation (7)
        return ⋃_{(k_1,...,k_ℓ) ∈ K_γ^π} ADDCONSTRAINTS(π[C_1⟨E_{1,k_1...k_ℓ}⟩, ..., C_n⟨E_{n,k_1...k_ℓ}⟩], E \ γ).
end
```

Consider for instance the class $\mathcal{C} = Av(B)$ for $B = \{1243, 2413, 531642, 41352\}$: $\mathcal{C}$ contains only one simple permutation (namely 3142), and $B^\star = \{1243\}$. Applying Algorithm 1 to this class $\mathcal{C}$ gives the following system of equations:

$$\begin{aligned}
\hat{\mathcal{C}}\langle 1243\rangle &= 1 \,\cup\, 12[\hat{\mathcal{C}}^+\langle 12\rangle, \hat{\mathcal{C}}\langle 132\rangle] \,\cup\, 12[\hat{\mathcal{C}}^+\langle 1243\rangle, \hat{\mathcal{C}}\langle 21\rangle] \,\cup\, 21[\hat{\mathcal{C}}^-\langle 1243\rangle, \hat{\mathcal{C}}\langle 1243\rangle] \\
&\cup\, 3142[\hat{\mathcal{C}}\langle 1243\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 21\rangle, \hat{\mathcal{C}}\langle 132\rangle] \,\cup\, 3142[\hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 132\rangle, \hat{\mathcal{C}}\langle 132\rangle] \qquad (8)\\
\hat{\mathcal{C}}\langle 12\rangle &= 1 \,\cup\, 21[\hat{\mathcal{C}}^-\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle] \qquad (9)\\
\hat{\mathcal{C}}\langle 132\rangle &= 1 \,\cup\, 12[\hat{\mathcal{C}}^+\langle 132\rangle, \hat{\mathcal{C}}\langle 21\rangle] \,\cup\, 21[\hat{\mathcal{C}}^-\langle 132\rangle, \hat{\mathcal{C}}\langle 132\rangle] \qquad (10)\\
\hat{\mathcal{C}}\langle 21\rangle &= 1 \,\cup\, 12[\hat{\mathcal{C}}^+\langle 21\rangle, \hat{\mathcal{C}}\langle 21\rangle]. \qquad (11)
\end{aligned}$$

5 Disambiguation of the system

In the above, Equation (8) gives an ambiguous description of the class $\hat{\mathcal{C}}\langle 1243\rangle$. As noticed in Albert and Atkinson (2005), we can derive an unambiguous equation using the inclusion-exclusion principle: $\hat{\mathcal{C}}\langle 1243\rangle = 1 \cup 12[\hat{\mathcal{C}}^+\langle 12\rangle, \hat{\mathcal{C}}\langle 132\rangle] \cup 12[\hat{\mathcal{C}}^+\langle 1243\rangle, \hat{\mathcal{C}}\langle 21\rangle] \setminus 12[\hat{\mathcal{C}}^+\langle 12\rangle, \hat{\mathcal{C}}\langle 21\rangle] \cup 21[\hat{\mathcal{C}}^-\langle 1243\rangle, \hat{\mathcal{C}}\langle 1243\rangle] \cup 3142[\hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 132\rangle, \hat{\mathcal{C}}\langle 132\rangle] \cup 3142[\hat{\mathcal{C}}\langle 1243\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 21\rangle, \hat{\mathcal{C}}\langle 132\rangle] \setminus 3142[\hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 21\rangle, \hat{\mathcal{C}}\langle 132\rangle]$. The system so obtained contains negative terms in general. This still gives a system of equations allowing to compute the generating function of the class. However, this cannot be easily used for random generation, as the subtraction of combinatorial objects is not handled by random samplers. In this section we disambiguate this system to obtain a new positive one: the key idea is to replace the negative terms by *complement sets*, hereby transforming pattern avoidance constraints into pattern *containment* constraints.

5.1 General framework

The starting point of the disambiguation is to rewrite ambiguous terms like $A \cup B \cup C$ as a disjoint union $(A \cap B \cap C) \uplus (\bar{A} \cap B \cap C) \uplus (\bar{A} \cap \bar{B} \cap C) \uplus (\bar{A} \cap B \cap \bar{C}) \uplus (A \cap \bar{B} \cap C) \uplus (A \cap \bar{B} \cap \bar{C}) \uplus (A \cap B \cap \bar{C})$. By disambiguating the union $A \cup B \cup C$ using complement sets instead of negative terms, we obtain an unambiguous description of the union with only positive terms. But when taking the complement of a set defined by pattern avoidance constraints, these are transformed into pattern *containment* constraints.

Therefore, for any set $\mathcal{P}$ of permutations, we define the *restriction* $\mathcal{P}\langle E\rangle(A)$ of $\mathcal{P}$ as the set of permutations that belong to $\mathcal{P}$ and that avoid every pattern of E and contain every pattern of A. This notation will be used when $\mathcal{P} = \hat{\mathcal{C}}^\varepsilon$, for $\varepsilon \in \{\ \ , +, -\}$ and $\mathcal{C}$ a permutation class. With this notation, notice also that for $A = \emptyset$, $\mathcal{C}\langle E\rangle = \mathcal{C}\langle E\rangle(\emptyset)$ is a standard permutation class. Restrictions have the nice feature of being stable by intersection as $\mathcal{P}\langle E\rangle(A) \cap \mathcal{P}\langle E'\rangle(A') = \mathcal{P}\langle E \cup E'\rangle(A \cup A')$. We also define a *restriction term* to be a set of permutations described as $\pi[\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n]$ where π is a simple permutation or 12 or 21 and the $\mathcal{S}_i$ are restrictions. By uniqueness of the substitution decomposition of a permutation, restriction terms are stable by intersection as well and the intersection is performed componentwise for terms sharing the same root: $\pi[\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n] \cap \pi[\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n] = \pi[\mathcal{S}_1 \cap \mathcal{T}_1, \mathcal{S}_2 \cap \mathcal{T}_2, \dots, \mathcal{S}_n \cap \mathcal{T}_n]$.

5.2 Disambiguate

The disambiguation of the system obtained by Algorithm 1 is performed by Algorithm 2. It consists in two main operations. One is the disambiguation of an equation according to the root of the terms that induce ambiguity, which may introduce right-only restrictions. This leads to the second procedure which computes new equations (that are added to the system) to describe these new restrictions (Algorithm 3).

Algorithm 2: DISAMBIGUATESYSTEM($\mathcal{E}$)

Data: A ambiguous system $\mathcal{E}$ of combinatorial equations /* obtained by Algo. 1 */
Result: An unambiguous system of combinatorial equations equivalent to $\mathcal{E}$
begin
  while *there is an ambiguous equation F in $\mathcal{E}$* **do**
    Take π a root that appears several times in F in an ambiguous way
    Replace the restriction terms of F whose root is π by a disjoint union using Eq. (12) – (14)
    while *there exists a right-only restriction $\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$ in some equation of $\mathcal{E}$* **do**
      $\mathcal{E} \longleftarrow \mathcal{E} \bigcup$ COMPUTEEQNFORRESTRICTION($\hat{\mathcal{C}}^{\varepsilon}$,$E$,$A$). /* See Algo. 3 */
  return $\mathcal{E}$
end

As stated in Section 4, every equation F of our system can be written as $t = 1 \cup t_1 \cup t_2 \cup t_3 \ldots \cup t_k$ where the t_i are restriction terms and t is a restriction. By uniqueness of the substitution decomposition of a permutation, terms of this union which have different roots π are disjoint. Thus for an equation we only need to disambiguate unions of terms with same root.

For example in Equation (8), there are two pairs of ambiguous terms which are terms with root 3142 and terms with root 12. Every ambiguous union can be written in the following unambiguous way:

$$\bigcup_{i=1}^{k} t_i = \biguplus_{X\subseteq[1\ldots k],X\neq\emptyset} \bigcap_{i\in X} t_i \cap \bigcap_{i\in \overline{X}} \overline{t_i}, \tag{12}$$

where the *complement* $\overline{t_i}$ of a restriction term t_i is defined as the set of permutations of $\hat{\mathcal{C}}$ whose decomposition tree has the same root than t_i but that do not belong to t_i. Equation 13 below shows that $\overline{t_i}$ is not a term in general but can be expressed as a disjoint union of terms. By distributivity of $\cap$ over $\uplus$, the above expression can therefore be rewritten as a disjoint union of intersection of terms. Because terms are stable by intersection, the right-hand side of Equation 12 is hereby written as a disjoint union of terms.

For instance, consider terms with root 3142 in Equation (8): $t_1 = 3142[\hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 132\rangle, \hat{\mathcal{C}}\langle 132\rangle]$ and $t_2 = 3142[\hat{\mathcal{C}}\langle 1243\rangle, \hat{\mathcal{C}}\langle 12\rangle, \hat{\mathcal{C}}\langle 21\rangle, \hat{\mathcal{C}}\langle 132\rangle]$. Equation (12) applied to t_1 and t_2 gives an expression of the form $\hat{\mathcal{C}}\langle 1243\rangle = 1 \cup 12[\ldots] \cup 12[\ldots] \cup 21[\ldots] \cup (t_1 \cap t_2) \uplus (t_1 \cap \overline{t_2}) \uplus (\overline{t_1} \cap t_2)$.

To compute the complement of a term t, it is enough to write that

$$\bar{t} = \biguplus_{X\subseteq\{1,\ldots,n\},X\neq\emptyset} \pi[\mathcal{S}'_1, \ldots, \mathcal{S}'_n] \text{ where } \mathcal{S}'_i = \overline{\mathcal{S}_i} \text{ if } i \in X \text{ and } \mathcal{S}'_i = \mathcal{S}_i \text{ otherwise,} \tag{13}$$

with the convention that $\overline{\mathcal{S}_i} = \hat{\mathcal{C}}^{\varepsilon} \setminus \mathcal{S}_i$ for $\mathcal{S}_i = \hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$. Indeed, by uniqueness of substitution decomposition, the set of permutations of $\hat{\mathcal{C}}$ that do not belong to t but whose decomposition tree has root π can be written as the union of terms $u = \pi[\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_n]$ where $\mathcal{S}'_i = \mathcal{S}_i$ or $\mathcal{S}'_i = \overline{\mathcal{S}_i}$ and at least one restriction $\mathcal{S}_i$ must be complemented. For example $\overline{21[\mathcal{S}_1, \mathcal{S}_2]} = 21[\mathcal{S}_1, \overline{\mathcal{S}_2}] \uplus 21[\overline{\mathcal{S}_1}, \mathcal{S}_2] \uplus 21[\overline{\mathcal{S}_1}, \overline{\mathcal{S}_2}]$.

The complement operation being pushed from restriction terms down to restrictions, we now compute $\overline{\mathcal{S}}$, for a given restriction $\mathcal{S} = \hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$, $\overline{\mathcal{S}}$ denoting the set of permutations of $\hat{\mathcal{C}}^{\varepsilon}$ that are not in $\mathcal{S}$. Notice that given a permutation σ of A, then any permutation τ of $\hat{\mathcal{C}}^{\varepsilon}\langle\sigma\rangle$ is in $\overline{\mathcal{S}}$ because τ avoids σ whereas permutations of $\mathcal{S}$ must contain σ. Symmetrically, if a permutation σ is in E then permutations of $\hat{\mathcal{C}}^{\varepsilon}\langle\rangle(\sigma)$ are in $\overline{\mathcal{S}}$. It is straightforward to check that $\overline{\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)} = \left[\bigcup_{\sigma\in E} \hat{\mathcal{C}}^{\varepsilon}\langle\rangle(\sigma)\right] \bigcup \left[\bigcup_{\sigma\in A} \hat{\mathcal{C}}^{\varepsilon}\langle\sigma\rangle()\right]$.

Unfortunately this expression is ambiguous. Like before we can rewrite it as an unambiguous union

$$\overline{\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)} = \biguplus_{\substack{X\subseteq A, Y\subseteq E\\ X\times Y\neq\emptyset\times\emptyset}} \hat{\mathcal{C}}^{\varepsilon}\langle X\cup\overline{Y}\rangle(Y\cup\overline{X}), \text{ where } \overline{X} = A\setminus X \text{ and } \overline{Y} = E\setminus Y. \tag{14}$$

In our example (Equations (8) to (11)), only trivial complements appear as every restriction is of the form $\hat{\mathcal{C}}\langle\sigma\rangle()$ or $\hat{\mathcal{C}}\langle\rangle(\sigma)$ for which complements are respectively $\hat{\mathcal{C}}\langle\rangle(\sigma)$ and $\hat{\mathcal{C}}\langle\sigma\rangle()$.

All together, for any equation of our system, we are able to rewrite it unambiguously as a disjoint union of restriction terms. As noticed before, some new right-only restrictions may appear during this process, for example as the result of the intersection of several restrictions or when complementing restrictions. To obtain a complete system we must compute iteratively equations defining these new restrictions using Algorithm 3 described below.

Finally, the terminaison of Algorithm 2 is easily proved. Indeed, for all the restrictions $\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$ that are considered in the inner loop of Algorithm 2, every permutation in the sets E and A is a pattern of some element of the basis B of $\mathcal{C}$. And since B is finite, there is a finite number of such restrictions.

5.3 Compute an equation for a restriction

Let $\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$ be a restriction. Our goal here is to find a combinatorial specification of this restriction in terms of smaller restriction terms (smaller w.r.t. inclusion).

If $A = \emptyset$, this is exactly the problem addressed in Section 4.2 and solved by pushing down the pattern avoidance constraints in the procedure ADDCONSTRAINTS of Algorithm 1. Algorithm 3 below shows how to propagate also the pattern *containment* constraints induced by $A \neq \emptyset$.

Algorithm 3: COMPUTEEQNFORRESTRICTION($\hat{\mathcal{C}}^{\varepsilon}, E, A$)

Data: $\hat{\mathcal{C}}^{\varepsilon}, E, A$ with E, A sets of permutations, $\hat{\mathcal{C}}^{\varepsilon}$ given by $\mathcal{S}_{\hat{\mathcal{C}}}$ and $\varepsilon \in \{\ , +, -\}$.
Result: An equation defining $\hat{\mathcal{C}}^{\varepsilon}\langle E\rangle(A)$ as a union of restriction terms.
begin
  $F \leftarrow$ Equation (1) or (2) or (3) (depending on ε)
  foreach $\sigma \in E$ **do**
    /* This step modifies F! */
    Replace any restriction term t in F by ADDCONSTRAINTS($t, \{\sigma\}$)   /* See Algo. 1 */
  foreach $\sigma \in A$ **do**
    /* This step modifies F! */
    Replace any restriction term t in F by ADDMANDATORY(t, σ)
  return F
end

ADDMANDATORY ($\pi[\mathcal{S}_1, \dots, \mathcal{S}_n], \gamma$)
  return a rewriting of $\pi[\mathcal{S}_1, \dots, \mathcal{S}_n](\gamma)$ as a union of restriction terms using Equation (15).
end

The pattern *containment* constraints are propagated by ADDMANDATORY, in a very similar fashion to the pattern *avoidance* constraints propagated by ADDCONSTRAINTS. To compute $t(\gamma)$ for γ a permutation and $t = \pi[\mathcal{S}_1, \dots, \mathcal{S}_n]$ a restriction term, we first compute all embeddings of γ into π. In this case, a permutation belongs to $t(\gamma)$ if and only if at least one embedding is satisfied. Hence, any restriction term $t = \pi[\mathcal{S}_1, \ldots, \mathcal{S}_n](\gamma)$ rewrites as a (possibly ambiguous) union as follows:

$$\bigcup_{i=1}^{\ell} \pi[\mathcal{S}_1(\gamma_{\alpha_i(1)}), \mathcal{S}_2(\gamma_{\alpha_i(2)}), \ldots, \mathcal{S}_n(\gamma_{\alpha_i(n)})], \tag{15}$$

where the $(\alpha_i)_{i\in\{1,\ldots,\ell\}}$ are all the embeddings of γ in π and if $\gamma_{\alpha_i(j)} = 0$, then $\mathcal{S}_j(\gamma_{\alpha_i(j)}) = \mathcal{S}_j$. For instance, for $t = 2413[\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4]$ and $\gamma = 3214$, there are 9 embeddings of γ into 2413, and the embedding $2413\{321, 1, 0, 0\}$ contributes to the above union with the term $2413[\mathcal{S}_1(321), \mathcal{S}_2(1), \mathcal{S}_3, \mathcal{S}_4]$.

Notice that although the unions of Equation 15 may be ambiguous, they will be transformed into disjoint unions by the outer loop of Algorithm 2. Finally, the algorithm produces an unambiguous system which is the result of a finite number of iterations of computing equations followed by their disambiguation.

6 Conclusion

We provide an algorithm to compute a combinatorial specification for a permutation class $\mathcal{C} = Av(B)$, when its basis B and the set of its simple permutations are finite and given as input. The complexity of this algorithm is however still to analyse. In particular, we observe a combinatorial explosion of the number of equations in the system obtained, that needs to be quantified.

Combined with existing algorithms, our procedure provides a full algorithmic chain from the basis (when finite) of a permutation class $\mathcal{C}$ to a specification for $\mathcal{C}$. This procedure may fail to compute its result, when $\mathcal{C}$ contains an infinite number of simple permutations, this condition being tested algorithmically.

This procedure has two natural algorithmic continuations. First, with the *dictionnary* of Flajolet and Sedgewick (2009), the constructors in the specification of $\mathcal{C}$ can be directly translated into operators on the generating function $C(z)$ of $\mathcal{C}$, turning the specification into a system of (possibly implicit) equations defining $C(z)$. Notice that, using the inclusion-exclusion principle as in Albert and Atkinson (2005), a system defining $C(z)$ could also be obtained from an *ambiguous* system describing $\mathcal{C}$. Second, the specification can be translated directly into a Boltzmann uniform random sampler of permutations in $\mathcal{C}$, in the same fashion as the above dictionnary (see Duchon et al. (2004)). This second translation is possible only from an unambiguous system: indeed, whereas adapted when considering enumeration sequences, the inclusion-exclusion principle does not apply when working on the combinatorial objects themselves.

When generating permutations with a Boltzmann sampler, complexity is measured w.r.t. the size of the permutation produced (and is linear if we allow a small variation on the size of the output permutation; quadratic otherwise) and not at all w.r.t. the number of equations in the specification. In our context, this dependency is of course relevant, and opens a new direction in the study of Boltzmann random samplers.

With a complete implementation of the algorithmic chain from B to the specification and the Boltzmann sampler, one should be able to test conjectures on and study permutation classes. One direction would be to somehow measure the randomness of permutations in a given class, by comparing random permutations with random permutations in a class, or random permutations in two different classes, w.r.t. well-known statistics on permutations. Another perspective would be to use the specifications obtained to compute or estimate the growth rates of permutation classes, to provide improvements on the known bounds on these growth rates. We could also explore the possible use the computed specifications to provide more efficient algorithms to test membership of a permutation to a class.

However, a weekness of our procedure that we must acknowledge is that it 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. By Albert and Atkinson (2005), such classes have finite basis (which is a restriction we imposed already), but they also have an *algebraic* generating function. Of course, this is not the case for every permutation class. We may wonder how restrictive this framework is, depending on which problems are studied. First, does it often happen that a permutation class contains finitely many simple permutations? To properly express what *often* means, a probability distribution on permutation classes should be defined, which is a direction of research yet to be explored. Second, we may want to describe some problems (maybe like the distribution of some statistics) for which algebraic permutation classes are representative of all permutation classes.

To enlarge the framework of application of our algorithm, we could explore the possibility of extending it to permutation classes that contain an infinite number of simple permutations, but that are finitely described (like the family of oscillations of Brignall et al. (2008b) for instance). With such an improvement, more classes would enter our framework, but it would be hard to leave the algebraic case. This is however a promising direction for the construction of Boltzmann random samplers for such permutation classes.

References

M. H. Albert and M. D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Math.*, 300 (1-3):1–15, 2005.

M. H. Albert, M. D. Atkinson, and M. Klazar. The enumeration of simple permutations. *J. Integer Seq.*, 6, 2003.

M. H. Albert, S. Linton, and N. Ruškuc. The insertion encoding of permutations. *Electron. J. Combin.*, 12:Research Paper 47, 31 pp. (electronic), 2005.

F. Bassino, M. Bouvel, A. Pierrot, and D. Rossin. Deciding the finiteness of simple permutations contained in a wreath-closed class is polynomial. *Pure Mathematics and Applications*, 21(2):119–135, 2010.

F. Bassino, M. Bouvel, A. Pierrot, and D. Rossin. A polynomial algorithm for deciding the finiteness of the number of simple permutations contained in permutation classes. Preprint available at `http://lipn.fr/~bassino/publications.html`, 2011.

P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65:277–283, 1998.

M. Bousquet-Mélou. Four classes of pattern-avoiding permutations under one roof: Generating trees with two labels. *Electron. J. Combin.*, 9(2), 2002.

R. Brignall. A survey of simple permutations. In S. Linton, N. Ruškuc, and V. Vatter, editors, *Permutation Patterns*, volume 376 of *London Math. Soc. Lecture Note Ser.*, pages 41–65. Cambridge Univ. Press, 2010.

R. Brignall, S. Huczynska, and V. Vatter. Simple permutations and algebraic generating functions. *J. Combin. Theory Ser. A*, 115(3):423–441, 2008a.

R. Brignall, N. Ruškuc, and V. Vatter. Simple permutations: decidability and unavoidable substructures. *Theoret. Comput. Sci.*, 391(1-2):150–163, 2008b.

P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Comb. Probab. Comput.*, 13(4–5):577–625, 2004.

S. Elizalde. *Statistics on pattern-avoiding permutations*. PhD thesis, MIT, 2004.

P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

S. Kitaev and T. Mansour. A survey on certain pattern problems. Preprint available at `http://www.ru.is/kennarar/sergey/index_files/Papers/survey.ps`, 2003.

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

A. Pierrot and D. Rossin. Simple permutation poset. Preprint available at `http://arxiv.org/abs/1201.3119`, 2012.

V. Vatter. Enumeration schemes for restricted permutations. *Comb. Probab. Comput.*, 17(1):137–159, 2008.