COMBINATORIAL GENERATION VIA PERMUTATION LANGUAGES. I. FUNDAMENTALS

ELIZABETH HARTUNG, HUNG P. HOANG, TORSTEN MÜTZE, AND AARON WILLIAMS

Abstract. In this work we present a general and versatile algorithmic framework for exhaustively generating a large variety of different combinatorial objects, based on encoding them as permutations. This approach provides a unified view on many known results and allows us to prove many new ones. In particular, we obtain the following four classical Gray codes as special cases: the Steinhaus-Johnson-Trotter algorithm to generate all permutations of an n-element set by adjacent transpositions; the binary reflected Gray code to generate all n-bit strings by flipping a single bit in each step; the Gray code for generating all n-vertex binary trees by rotations due to Lucas, van Baronaigien, and Ruskey; the Gray code for generating all partitions of an n-element ground set by element exchanges due to Kaye.

We present two distinct applications for our new framework: The first main application is the generation of pattern-avoiding permutations, yielding new Gray codes for different families of permutations that are characterized by the avoidance of certain classical patterns, (bi)vincular patterns, barred patterns, Bruhat-restricted patterns, mesh patterns, monotone and geometric grid classes, and many others. We also obtain new Gray codes for all the combinatorial objects that are in bijection to these permutations, in particular for five different types of geometric rectangulations, also known as floorplans, which are divisions of a square into n rectangles subject to certain restrictions.

The second main application of our framework are lattice congruences of the weak order on the symmetric group S_n. Recently, Pilaud and Santos realized all those lattice congruences as $(n-1)$-dimensional polytopes, called quotientopes, which generalize hypercubes, associahedra, permutahedra etc. Our algorithm generates the equivalence classes of each of those lattice congruences, by producing a Hamilton path on the skeleton of the corresponding quotientope, yielding a constructive proof that each of these highly symmetric graphs is Hamiltonian. We thus also obtain a provable notion of optimality for the Gray codes obtained from our framework: They translate into walks along the edges of a polytope.

(Elizabeth Hartung) Massachusetts College of Liberal Arts, United States

(Hung P. Hoang) Department of Computer Science, ETH Zürich, Switzerland

(Torsten Mütze) Department of Computer Science, University of Warwick, United Kingdom

(Aaron Williams) Computer Science Department, Williams College, United States

E-mail addresses: e.hartung@mcla.edu, hung.hoang@inf.ethz.ch, torsten.mutze@warwick.ac.uk, aaron.williams@williams.edu.

Key words and phrases. Exhaustive generation algorithm, Gray code, pattern-avoiding permutation, weak order, lattice congruence, quotientope, Hamilton path, rectangulation.

An extended abstract of this paper has been accepted for presentation at the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020).

Torsten Mütze is also affiliated with the Faculty of Mathematics and Physics, Charles University Prague, Czech Republic. He was supported by Czech Science Foundation grant GA 19-08554S, and by German Science Foundation grant 413902284.

1. Introduction

In mathematics and computer science we frequently encounter different kinds of combinatorial objects, such as permutations, binary strings, binary trees, set partitions, spanning trees of a graph, and so forth. There are three recurring fundamental algorithmic tasks that we want to perform with such objects: counting, random generation, and exhaustive generation. For the first two tasks, there are powerful general methods available, such as generating functions [FS09] and Markov chains [Jer03], solving both problems for a large variety of different objects. For the third task, namely exhaustive generation, however, we are lacking such a powerful and unifying theory, even though some first steps in this direction have been made (see Section 1.2 below). Nonetheless, the literature contains a vast number of algorithms that solve the exhaustive generation problem for specific classes of objects, and many of these algorithms are covered in depth in the most recent volume of Knuth's seminal series *'The Art of Computer Programming'* [Knu11].

1.1. Overview of our results.

The main contribution of this paper is a general and versatile algorithmic framework for exhaustively generating a large variety of different combinatorial objects, which provides a unified view on many known results and allows us to prove many new ones. The basic idea is to encode a particular set of objects as a set of permutations $L_n \subseteq S_n$, where S_n denotes the set of all permutations of $[n] := \{1, 2, \ldots, n\}$, and to use a simple greedy algorithm to generate those permutations by cyclic rotations of substrings, an operation we call a *jump*. This works under very mild assumptions on the set L_n, and allows us to generate more than double-exponentially (in n) many distinct sets L_n. Moreover, the jump orderings obtained from our algorithm translate into listings of combinatorial objects where consecutive objects differ by small changes, i.e., we obtain *Gray codes* [Sav97], and those changes are smallest possible in a provable sense. The main tools of our framework are Algorithm J and Theorem 1 in Section 2. In particular, we obtain the following four classical Gray codes as special cases: (1) the Steinhaus-Johnson-Trotter algorithm to generate all permutations of $[n]$ by adjacent transpositions, also known as plain change order [Tro62, Joh63]; (2) the binary reflected Gray code (BRGC) to generate all binary strings of length n by flipping a single bit in each step [Gra53]; (3) the Gray code for generating all n-vertex binary trees by rotations due to Lucas, van Baronaigien, and Ruskey [LvBR93]; (4) the Gray code for generating all set partitions of $[n]$ by exchanging an element in each step due to Kaye [Kay76].

We present two distinct applications for our new framework: The first main application is the generation of pattern-avoiding permutations, yielding new Gray codes for different families of permutations that are characterized by the avoidance of certain classical patterns, vincular and bivincular patterns [BS00, BMCDK10], barred patterns [Wes90], Bruhat-restricted patterns [WY06], mesh patterns [BC11], monotone and geometric grid classes [HV06, AAB$^+$13], and many others. We also obtain new Gray codes for all the combinatorial objects that are in bijection to these permutations, in particular for five different types of geometric rectangulations [ABP06, Rea12b, ABBM$^+$13, CSS18], also known as floorplans, which are divisions of a square into n rectangles subject to different restrictions (see Figure 5). Our main results in this area are summarized in Theorems 8 and 14 and Table 1 in Section 3, and in Theorem 21 in Section 4.

The second main application of our framework are lattice congruences of the weak order on the symmetric group S_n. This area has beautiful ramifications into groups, posets, polytopes, geometry, and combinatorics, and has been developed considerably in recent years, in particular thanks to Nathan Reading's works, summarized in [Rea12a, Rea16a, Rea16b]. There are double-exponentially many distinct such lattice congruences, and they generalize many known lattices such as the Boolean lattice, the Tamari lattice [Tam62], and certain Cambrian lattices [Rea06, CP17]. Recently, Pilaud and Santos [PS19] realized all those lattice congruences as $(n-1)$-dimensional polytopes, called quotientopes, which generalize hypercubes, associahedra, permutahedra etc. Our algorithm generates the equivalence classes of each of those lattice congruences, by producing a Hamilton path on the skeleton of the corresponding quotientope, yielding a constructive proof that each of these highly symmetric graphs is Hamiltonian; see Figure 11. Our results in this area are summarized in Theorem 22 and Corollary 23 in Section 5.

1.2. **Related work.** Avis and Fukuda [AF96] introduced *reverse-search* as a general technique for exhaustive generation. Their idea is to consider the set of objects to be generated as the nodes of a graph, and to connect them by edges that model local modification operations (for instance, adjacent transpositions for permutations). The resulting *flip graph* is equipped with an objective function, and the directed tree formed by the movements of a local search algorithm that optimizes this function is traversed backwards from the optimum node, using an adjacency oracle. The authors applied this technique successfully to derive efficient generation algorithms for a number of different objects; for instance, triangulations of a point set, spanning trees of a graph etc. Reverse-search is complementary to our permutation based approach, as both techniques use fundamentally different encodings of the objects. The permutation encoding seems to allow for more fine-grained control (optimal Gray codes) and even faster generation algorithms.

Another method for combinatorial counting and exhaustive generation is the *ECO framework* introduced by Barcucci, Del Lungo, Pergola, and Pinzani [BDLPP99]. The main tool is an infinite tree with integer node labels, and a set of production rules for creating the children of a node based on its label. Bacchelli, Barcucci, Grazzini, and Pergola [BBGP04] also used ECO for exhaustive generation, deriving an efficient algorithm for generating the corresponding root-to-node label sequences in the ECO tree in lexicographic order, which was later turned into a Gray code [BGPP07]. Dukes, Flanagan, Mansour, and Vajnovszki [DFMV08], Baril [Bar09], and Do, Tran and Vajnovszki [DTV19] used ECO for deriving Gray codes for different classes of pattern-avoiding permutations, which works under certain regularity assumptions on the production rules. Vajnovszki [Vaj10] also applied ECO for efficiently generating other classes of permutations, such as involutions and derangements. The main difference between ECO and our framework is that the change operations on the label sequences of the ECO tree do not necessarily correspond to Gray-code like changes on the corresponding combinatorial objects. Minimal jumps in a permutation, on the other hand, always correspond to minimal changes on the combinatorial objects in a provable sense, even though they may involve several entries of the permutation.

Li and Sawada [LS09] considered another tree-based approach for generating so-called *reflectable languages*, yielding Gray codes for k-ary strings and trees, restricted growth strings, and open meandric systems (see also [XCU10]). Ruskey, Sawada, and Williams [RSW12, SW12] proposed a generation framework based on binary strings with a fixed numbers of 1s, called *bubble languages*, which allows to generate e.g. combinations, necklaces, Dyck words, and Lyndon words. In the resulting cool-lex Gray codes, any two consecutive words differ by cyclic rotation of some prefix.

Pattern avoidance in permutations is a central topic in combinatorics, as illustrated by the books [Kit11, Bón12], and by the conference 'Permutation Patterns', held annually since 2003. Given two permutations π and τ, we say that π *contains the pattern* τ, if π contains a subpermutation formed by (not necessarily consecutive) entries that appear in the same relative order as in τ; otherwise we say that π *avoids* τ. It is well known that many fundamental classes of combinatorial objects are in bijection with pattern-avoiding permutations (see Table 1 and [Ten18]). For instance, Knuth [Knu98] first proved that all 123-avoiding and 132-avoiding permutations are counted by the Catalan numbers (see also [CK08]). With regards to counting and exhaustive generation, a few tree-based algorithms for pattern-avoiding permutations have been proposed [Eli07, DFMV08, Bar08, Bar09]. Pattern-avoidance has also been studied extensively from an algorithmic point of view. In fact, testing whether a permutation π contains another permutation τ as a pattern is known to be NP-complete in general [BBL98]. Jelínek and Kynčl [JK17] proved that the problem remains hard even if π and τ have no decreasing subsequence of length 4 and 3, respectively, which is best possible. On the other hand, Guillemot and Marx [GM14] showed that the problem can be solved in time $2^{O(k^2 \log k)} n \log n$, where n is the length of π and k is the length of τ, a considerable improvement over the obvious $O(n^k)$ algorithm (see also [Koz19]). It is also known that for a fixed set of forbidden patterns, computing the number of pattern-avoiding permutations is hard [GP16].

1.3. **Outline of this paper.** This is the first in a series of papers where we develop our theory of combinatorial generation via permutation languages. In this first paper we focus on presenting the fundamental algorithmic ideas (Section 2), and their main applications to pattern-avoiding permutations (Sections 3 and 4) and lattice congruences (Section 5). We present detailed results and proofs for pattern-avoiding permutations, while we only state the main results for lattice congruences. The proofs for our results on lattice congruences and a more detailed analysis of them will be given in part II of this series. In future parts, we will also cover efficient algorithms and rectangulations, important topics that can only be briefly scratched here due to the limited space (see Section 2.7 and Figure 5 below, respectively).

2. Generating permutations by jumps

In this section we present a simple greedy algorithm, Algorithm J, for exhaustively generating a given set $L_n \subseteq S_n$ of permutations, and we show that the algorithm works successfully under very mild assumptions on the set L_n (Theorem 1).

2.1. **Preliminaries.** We use S_n to denote the set of all permutations of $[n] := \{1, \ldots, n\}$, and we write $\pi \in S_n$ in one-line notation as $\pi = \pi(1)\pi(2)\ldots\pi(n) = a_1a_2\ldots a_n$. We use $\mathrm{id}_n = 12\ldots n$ to denote the identity permutation, and $\varepsilon \in S_0$ to denote the empty permutation. For any $\pi \in S_{n-1}$ and any $1 \leq i \leq n$, we write $c_i(\pi) \in S_n$ for the permutation obtained from π by inserting the new largest value n at position i of π, i.e., if $\pi = a_1 \ldots a_{n-1}$ then $c_i(\pi) = a_1 \ldots a_{i-1}\, n\, a_i \ldots a_{n-1}$. Moreover, for $\pi \in S_n$, we write $p(\pi) \in S_{n-1}$ for the permutation obtained from π by removing the largest entry n. Here, c_i and p stand for the child and parent of a node in the tree of permutations discussed shortly.

Given a permutation $\pi = a_1 \ldots a_n$ with a substring $a_i \ldots a_j$ with $a_i > a_{i+1}, \ldots, a_j$, a *right jump of a_i by $j-i$ steps* is a cyclic left rotation of this substring by one position to $a_{i+1} \ldots a_j a_i$. Similarly, given a substring $a_i \ldots a_j$ with $a_j > a_i, \ldots, a_{j-1}$, a *left jump of a_j by $j-i$ steps* is a cyclic right rotation of this substring to $a_j a_i \ldots a_{j-1}$. For example, a right jump of 5 in the permutation 265134 by 2 steps yields 261354.

2.2. **The basic algorithm.** Our approach starts with the following simple greedy algorithm to generate a set of permutations $L_n \subseteq S_n$. We say that a jump is *minimal* (w.r.t. L_n), if a jump of the same value in the same direction by fewer steps creates a permutation that is not in L_n. Note that each entry of the permutation admits at most one minimal left jump and at most one minimal right jump.

Algorithm J *(Greedy minimal jumps).* This algorithm attempts to greedily generate a set of permutations $L_n \subseteq S_n$ using minimal jumps starting from an initial permutation $\pi_0 \in L_n$.

J1. [Initialize] Visit the initial permutation π_0.

J2. [Jump] Generate an unvisited permutation from L_n by performing a minimal jump of the largest possible value in the most recently visited permutation. If no such jump exists, or the jump direction is ambiguous, then terminate. Otherwise visit this permutation and repeat J2.

Put differently, in step J2 we consider the entries $n, n-1, \ldots, 2$ of the current permutation in decreasing order, and for each of them we check whether it allows a minimal left or right jump that creates a previously unvisited permutation, and we perform the first such jump we find, unless the same entry also allows a jump in the opposite direction, in which case we terminate. If no minimal jump creates an unvisited permutation, we also terminate the algorithm prematurely. For example, consider $L_4 = \{1243, 1423, 4123, 4213, 2134\}$. Starting with $\pi_0 = 1243$, the algorithm generates $\pi_1 = 1423$ (obtained from π_0 by a left jump of 4 by 1 step), then $\pi_2 = 4123$, then $\pi_3 = 4213$ (in π_2, 4 cannot jump, as π_0 and π_1 have been visited before; 3 cannot jump either to create any permutation from L_4, so 2 jumps left by 1 step), then $\pi_4 = 2134$, successfully generating L_4. If instead we initialize with $\pi_0 = 4213$, then the algorithm generates $\pi_1 = 2134$, and then stops, as no further jump is possible. If we choose $\pi_0 = 1423$, then we may jump 4 to the left or right (by 1 step), but as the direction is ambiguous, the algorithm stops immediately. As mentioned before, the algorithm may stop prematurely only either because no minimal jump leading to a new permutation from L_n is possible, or because the direction of jump is ambiguous in some step. By the definition of step J2, the algorithm will never visit any permutation twice.

The following main result of our paper provides a sufficient condition on the set L_n to guarantee that Algorithm J is successful. This condition is captured by the following closure property of the set L_n. A set of permutations $L_n \subseteq S_n$ is called a *zigzag language*, if either $n = 0$ and $L_0 = \{\varepsilon\}$, or if $n \geq 1$ and $L_{n-1} := \{p(\pi) \mid \pi \in L_n\}$ is a zigzag language satisfying the following condition:

(z) For every $\pi \in L_{n-1}$ we have $c_1(\pi) \in L_n$ and $c_n(\pi) \in L_n$.

Theorem 1. *Given any zigzag language of permutations L_n and initial permutation $\pi_0 = \mathrm{id}_n$, Algorithm J visits every permutation from L_n exactly once.*

Remark 2. It is easy to see that the number of zigzag languages is at least $2^{(n-1)!(n-2)} = 2^{2^{\Theta(n \log n)}}$, i.e., it is more than double-exponential in n. We will see that many of these languages do in fact encode interesting combinatorial objects. Moreover, minimal jumps as performed by Algorithm J always translate to small changes on those objects in a provable sense, i.e., our algorithm defines Gray codes for a large variety of combinatorial objects, and Hamilton paths/cycles on the corresponding flip graphs and polytopes.

Before we present the proof of Theorem 1, we give two equivalent characterizations of zigzag languages.

2.3. **Characterization via the tree of permutations.** There is an intuitive characterization of zigzag languages via the *tree of permutations*. This is an infinite (unordered) rooted tree which has as nodes all permutations from S_n at distance n from the root; see Figure 1. Specifically, the empty permutation ε is at the root, and the children of any node $\pi \in S_{n-1}$ are exactly the permutations $c_i(\pi)$, $1 \leq i \leq n$, i.e., the permutations obtained by inserting the new largest value n in all possible positions. Consequently, the parent of any node $\pi' \in S_n$ is exactly the permutation $p(\pi')$ obtained by removing the largest value n. In the figure, for any node $\pi \in S_{n-1}$, the nodes

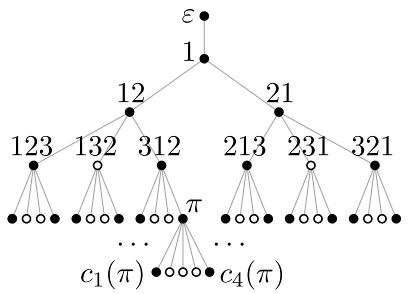


FIGURE 1. Tree of permutations, where the children $c_1(\pi)$ and $c_n(\pi)$ of any node $\pi \in S_{n-1}$ are drawn black, all others white.

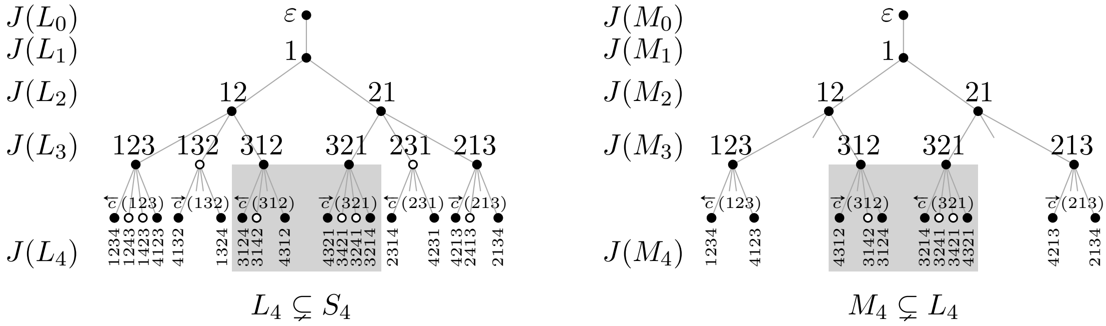


FIGURE 2. Ordered tree representation of two zigzag languages of permutations L_4 (left) and M_4 (right) with $M_4 \subsetneq L_4 \subsetneq S_4$. Both trees contain the same sets of permutations in the subtrees rooted at 312 and 321 (highlighted in gray), but in the corresponding sequences $J(L_4)$ and $J(M_4)$, those permutations appear in different relative order due to the node 132, which was pruned from the right tree.

representing the children $c_1(\pi)$ and $c_n(\pi)$ are drawn black, whereas the other children are drawn white. Any zigzag language of permutations can be obtained from this full tree by pruning subtrees, where by condition (z) a subtree may be pruned only if its root $\pi' \in S_n$ is neither the child $c_1(\pi)$ nor the child $c_n(\pi)$ of its parent $\pi = p(\pi') \in S_{n-1}$, i.e., only subtrees rooted at white nodes may be pruned. For any subtree obtained by pruning according to this rule and for any $n \geq 1$, the remaining permutations of length n form a zigzag language L_n; see Figure 2.

Consider all nodes in the tree for which the entire path to the root consists only of black nodes. Those nodes never get pruned and are therefore contained in any zigzag language. These are exactly all permutations without peaks. A *peak* in a permutation $a_1 \dots a_n$ is a triple $a_{i-1}a_ia_{i+1}$ with $a_{i-1} < a_i > a_{i+1}$, and the language of permutations without peaks is generated by the recurrence $P_0 := \{\varepsilon\}$ and $P_n := \{c_1(\pi), c_n(\pi) \mid \pi \in P_{n-1}\}$ for $n \geq 1$. It follows that we have $|P_n| = 2^{n-1}$ and $P_n \subseteq L_n \subseteq S_n$ for any zigzag language L_n, i.e., L_n is sandwiched between the language of permutations without peaks and between the language of all permutations.

2.4. **Characterization via nuts.** Given a permutation π, we may repeatedly remove the largest value from it as long as it is in the leftmost or rightmost position. The remaining permutation is called the *nut* of π. For example, given $\pi = 965214378$, we can remove $9, 8, 7, 6, 5$, yielding 2143 as the nut of π. A left or right jump of some value in a permutation is *maximum* if there is no left jump or right jump of the same value with more steps. For example, in $\pi = 965214378$ a maximum right jump of 6 gives $\pi' = 952143678$. By unrolling the recursive definition of zig-zag languages from before, we obtain that $L_n \subseteq S_n$ is a zigzag language if and only if for all $\pi \in L_n$ both the maximum left jump and the maximum right jump of the value i yield another permutation in L_n for all $k \leq i \leq n$, where k is the largest value in π's nut (with $k = 2$ if the nut is empty).

2.5. **Proof of Theorem 1.** Given a zigzag language L_n, we define a sequence $J(L_n)$ of all permutations from L_n, and we prove that Algorithm J generates the permutations of L_n exactly in this order. For any $\pi \in L_{n-1}$ we let $\overrightarrow{c}(\pi)$ be the sequence of all $c_i(\pi) \in L_n$ for $i = 1, 2, \ldots, n$, starting with $c_1(\pi)$ and ending with $c_n(\pi)$, and we let $\overleftarrow{c}(\pi)$ denote the reverse sequence, i.e., it starts with $c_n(\pi)$ and ends with $c_1(\pi)$. In words, those sequences are obtained by inserting into π the new largest value n in all possible positions from left to right, or from right to left, respectively. The sequence $J(L_n)$ is defined recursively as follows: If $n = 0$ then $J(L_0) := \varepsilon$, and if $n \geq 1$ then we consider the finite sequence $J(L_{n-1}) =: \pi_1, \pi_2, \ldots$ and define

$$J(L_n) := \overleftarrow{c}(\pi_1), \overrightarrow{c}(\pi_2), \overleftarrow{c}(\pi_3), \overrightarrow{c}(\pi_4), \ldots, \tag{1}$$

i.e., this sequence is obtained from the previous sequence by inserting the new largest value n in all possible positions alternatingly from right to left, or from left to right; see Figure 2.

Remark 3. Algorithm J thus defines a left-to-right ordering of the nodes at distance n of the root in the tree representation of the zigzag language L_n described before, and this ordering is captured by the sequence $J(L_n)$; see Figure 2. Clearly, the same is true for all the zigzag languages $L_0, L_1, \ldots, L_{n-1}$ that are induced by L_n through the rule $L_{k-1} := \{p(\pi) \mid \pi \in L_k\}$ for $k = n, n-1, \ldots, 1$. The unordered tree is thus turned into an ordered tree, and it is important to realize that pruning operations change the ordering. Specifically, given two zigzag languages L_n and M_n with $M_n \subseteq L_n$, the tree for M_n is obtained from the tree for L_n by pruning, but in general $J(M_n)$ is *not* a subsequence of $J(L_n)$, as shown by the example in Figure 2. This shows that our approach is quite different from the one presented by Vajnovszki and Vernay [VV11], which considers only subsequences of the Steinhaus-Johnson-Trotter order $J(S_n)$.

Proof of Theorem 1. For any $\pi \in L_n$, we let $J(L_n)_\pi$ denote the subsequence of $J(L_n)$ that contains all permutations up to and including π. An immediate consequence of the definition of zigzag language is that L_n contains the identity permutation $\mathrm{id}_n = c_n(\mathrm{id}_{n-1})$. Moreover, the definition (1) implies that id_n is the very first permutation in the sequence $J(L_n)$.

We now argue by double induction over n and the length of $J(L_n)$ that Algorithm J generates all permutations from L_n exactly in the order described by the sequence $J(L_n)$, and that when we perform a minimal jump with the largest possible value to create a previously unvisited permutation, then there is only one direction (left or right) to which it can jump. The induction basis $n = 0$ is clear. Now suppose the claim holds for the zigzag language $L_{n-1} := \{p(\pi) \mid \pi \in L_n\}$. We proceed to show that it also holds for L_n.

As argued before, the identity permutation id_n is the first permutation in the sequence $J(L_n)$, and this is indeed the first permutation visited by Algorithm J in step J1. Now let $\pi \in L_n$ be the permutation currently visited by the algorithm in step J2, and let $\pi' := p(\pi) \in L_{n-1}$. If π' appears at an odd position in $J(L_{n-1})$, then we define $\bar{c} := \overleftarrow{c}(\pi')$ and otherwise we define $\bar{c} := \overrightarrow{c}(\pi')$. By (1), we know that π appears in the subsequence $\bar{c}$ within $J(L_n)$. We first consider the case that π is not the last permutation in $\bar{c}$. In this case, the permutation ρ succeeding π in $J(L_n)$ is obtained from π by a minimal jump (w.r.t. L_n) of the largest value n in some direction d, which is left if $\bar{c} = \overleftarrow{c}(\pi')$ and right if $\bar{c} = \overrightarrow{c}(\pi')$. Now observe that by the definition of $\bar{c}$, all permutations in L_n obtained from π by jumping n in the direction opposite to d precede π in $J(L_n)$ and have been visited by Algorithm J before by induction. Consequently, to generate a previously unvisited permutation, the value n can only jump in direction d in step J2 of the algorithm. Again by the definition of $\bar{c}$, the permutation ρ is obtained from π by a minimal jump (w.r.t. L_n), so the next permutation generated by the algorithm will indeed be ρ.

It remains to consider the case that π is the last permutation in the subsequence $\bar{c}$ within $J(L_n)$. Let ρ' be the permutation suceeding π' in $J(L_{n-1})$. By induction, we have the following property (*): ρ' is obtained from π' by a minimal jump (w.r.t. L_{n-1}) of the largest possible value a by k steps in some direction d (left or right), and a can jump only into one direction. As π is the last permutation in $\bar{c}$, the largest value n of π is at the boundary, which is the left boundary if $\bar{c} = \overleftarrow{c}(\pi')$ or the right boundary if $\bar{c} = \overrightarrow{c}(\pi')$. By (1), the permutation ρ succeeding π in $J(L_n)$ also has n at the same boundary, i.e., ρ differs from π by a jump of a by k steps in direction d. Suppose for the sake of contradiction that when transforming the currently visited permutation π in step J2, the algorithm does not perform this jump operation, but another one. This could be a jump of a larger value $b > a$ to transform π into some permutation $\tau \in L_n$ that is different from ρ and not in $J(L_n)_\pi$, or a jump of a in the direction opposite to d, or a jump of a in direction d by fewer than k steps. But in all those cases the permutation $\tau' := p(\tau) \in L_{n-1}$ is different from ρ' and not in $J(L_{n-1})_{\pi'}$, and it is obtained from π' by a jump of $b > a$, or a jump of a in the direction opposite to d, or a jump of a in direction d by fewer than k steps, respectively, a contradiction to property (*). This completes the proof. ☐

2.6. **Further properties of Algorithm J.** The next lemma captures when the algorithm generates a *cyclic* listing of permutations.

Lemma 4. *In the ordering of permutations $J(L_n)$ generated by Algorithm J, the first and last permutation are related by a minimal jump if and only if $|L_k|$ is even for all $2 \leq k \leq n-1$.*

Proof. Let π_k be the last permutation in the ordering $J(L_k)$ for all $k = 0, 1, \ldots, n$. For $k \geq 1$, we see from (1) that $\pi_k = c_k(\pi_{k-1})$ if $|L_{k-1}|$ is even and $\pi_k = c_1(\pi_{k-1})$ if $|L_{k-1}|$ is odd. As $|L_1| = 1$ is odd, we know that 1 and 2 are reversed in π_n, and so all numbers $|L_k|$, $2 \leq k \leq n-1$, must be even for id_n and π_n to be related by a minimal jump. ☐

Remark 5. It follows from the proof of Theorem 1 that instead of initializing the algorithm with the identity permutation $\pi_0 = \mathrm{id}_n$, we may use any permutation without peaks as a seed π_0.

2.7. **Efficiency considerations.** Let us make it very clear that in its stated form, Algorithm J is not an efficient algorithm to actually generate a particular zigzag language of permutations. The reason is that it requires storing the list of all previously visited permutations in order to decide which one to generate next. However, by introducing a few additional arrays, the algorithm can be made memoryless, so that such lookup operations are not needed anymore, and hence no permutations need to be stored at all. The efficiency of the resulting algorithm is then only determined by the efficiency with which we are able to compute minimal jumps with respect to the input zigzag language L_n for a given entry of the permutation. This leads to an algorithm that computes the next permutation to be visited in polynomial time. In many cases, this can be improved to a loopless algorithm that generates each new permutation in constant worst-case time. The key insight here is that any jump changes the inversion table of a permutation only in a single entry. By maintaining only the inversion table, jumps can thus be performed efficiently, even if the number of steps is big. This extensive and important discussion, however, is not the main focus here, and is deferred to a future part of this paper series.

2.8. **A general recipe.** Here is a step-by-step approach to apply our framework to the generation of a given family X_n of combinatorial objects. The first step is to establish a bijection f that encodes the objects from X_n as permutations $L_n \subseteq S_n$. If L_n is a zigzag language, which can be checked by verifying the closure property, then we may run Algorithm J with input L_n, and interpret the resulting ordering $J(L_n)$ in terms of the combinatorial objects, by applying f^{-1} to each permutations in $J(L_n)$, yielding an ordering on X_n. We may also apply f^{-1} to Algorithm J directly, which will yield a simple greedy algorithm for generating X_n. The final step is to make these algorithms efficient, by introducing additional data structures that allow the change operations on X_n (which are the preimages of minimal jumps under f) as efficiently as possible.

Let us illustrate these steps for the set X_n of binary strings of length $n-1$. We map any binary string $x = x_2 \dots x_n$ to a permutation $f(x) \in S_n$ by setting $f(\varepsilon) := 1$ and

$$f(x_2 \dots x_n) := \begin{cases} c_n(f(x_2 \dots x_{n-1})) & \text{if } x_n = 0, \\ c_1(f(x_2 \dots x_{n-1})) & \text{if } x_n = 1, \end{cases}$$

i.e., we build the permutation $f(x)$ by inserting the values $i = 2, \dots, n$ one by one, either at the leftmost or rightmost position, depending on the bit x_i. Observe that $f(X_n)$ is exactly the set of permutations without peaks $P_n \subseteq S_n$ discussed in Section 2.3 before, and a jump of the entry i in the permutation translates to flipping the bit x_i. Moreover, $f^{-1}(J(P_n))$ is exactly the well-known reflected Gray code BRGC for binary strings of length $n-1$ [Gra53], which can be implemented efficiently [BER76]. Applying f^{-1} to Algorithm J yields the following simple greedy algorithm for generating the BRGC (see [Wil13]): **J1.** Visit the initial all-zero string. **J2.** Repeatedly flip the rightmost bit that yields a previously unvisited string.

3. Pattern-avoiding permutations

The first main application of our framework is the generation of pattern-avoiding permutations. Our main results in this section are summarized in Theorem 8, Theorem 14 (and its corollaries Lemmas 9–13), and in Table 1. We emphasize that all our results can be generalized to bounding the number of appearances of patterns, where the special case with a bound of 0 appearances is pattern-avoidance; see Section 3.9 below.

3.1. Preliminaries.

The following simple but powerful lemma follows immediately from the definition of zigzag languages given in Section 2. For any set $L_n \subseteq S_n$ we define $p(L_n) := \{p(\pi) \mid \pi \in L_n\}$.

Lemma 6. *Let $L_n, M_n \subseteq S_n$, $n \geq 1$, be two zigzag languages of permutations. Then $L_n \cup M_n$ and $L_n \cap M_n$ are also zigzag languages of permutations, and we have $p(L_n \cup M_n) = p(L_n) \cup p(M_n)$ and $p(L_n \cap M_n) = p(L_n) \cap p(M_n)$.*

We say that two sequences of integers σ and τ are *order-isomorphic*, if their elements appear in the same relative order in both sequences. For instance, 2576 and 1243 are order-isomorphic. Given two permutations $\pi \in S_n$ and $\tau \in S_k$, we say that π *contains the pattern* τ, if and only if $\pi = a_1 \dots a_n$ contains a subpermutation $a_{i_1} \dots a_{i_k}$, $i_1 < \dots < i_k$, that is order-isomorphic to τ. We refer to such a subpermutation as a *match* of τ in π. If π does not contain the pattern τ, then we say that π *avoids* τ. For example, $\pi = 635412$ contains the pattern $\tau = 231$, as the highlighted entries form a match of τ in π. On the other hand, $\pi = 654123$ avoids $\tau = 231$. For any permutation τ, we let $S_n(\tau)$ denote all permutations from S_n avoiding the pattern τ. For propositional formulas F and G consisting of logical ANDs $\wedge$, ORs $\vee$, and patterns as variables, we define

$$\begin{aligned} S_n(F \wedge G) &:= S_n(F) \cap S_n(G), \\ S_n(F \vee G) &:= S_n(F) \cup S_n(G). \end{aligned} \tag{2}$$

For instance, $S_n(\tau_1 \wedge \cdots \wedge \tau_\ell)$ is the set of permutations avoiding each of the patterns $\tau_1, \ldots, \tau_\ell$, and $S_n(\tau_1 \vee \cdots \vee \tau_\ell)$ is the set of permutations avoiding at least one of the patterns $\tau_1, \ldots, \tau_\ell$.

Remark 7. From the point of view of counting, we clearly have $|L_n \cup M_n| = |L_n| + |M_n| - |L_n \cap M_n|$, so the problem of counting the union of two zigzag languages can be reduced to counting the individual languages and the intersection. However, from the point of view of exhaustive generation, we clearly do not want to take this approach, namely generate all permutations in L_n, all permutations in M_n, all permutations in $L_n \cap M_n$, and then combine and reduce those lists. This shows that the problem of generating languages like $S_n(\tau_1 \vee \cdots \vee \tau_\ell)$ or $S_n(F)$ for more general formulas F is genuinely interesting in our context. We will see a few applications of this general setting below, and we feel that this direction of generalization deserves further investigation by the pattern-avoidance community.

3.2. **Tame patterns.** We say that an infinite sequence of sets $L_0, L_1, \ldots$ is *hereditary*, if $L_{i-1} = p(L_i)$ holds for all $i \geq 1$. We say that a permutation pattern τ is *tame*, if $S_n(\tau)$, $n \geq 0$, is a hereditary sequence of zigzag languages. The hereditary property ensures that for a given set $S_n(\tau) =: L_n$, we can check membership within the families $L_{i-1} := p(L_i)$ for $i = n, n-1, \ldots, 1$ simply by checking for matches of the pattern τ. In terms of the aforementioned tree representation of zigzag languages, it means that all sets $S_n(\tau)$, $n \geq 0$, arise from pruning the infinite rooted tree of permutations in the same way (and not in different ways for the same pattern and different values of n), by considering the infinite sequence of node sets remaining in each level.

The following theorem is an immediate consequence of Lemma 6 and the definition (2).

Theorem 8. *Let F be an arbitrary propositional formula consisting of logical ANDs $\wedge$, ORs $\vee$, and tame patterns as variables, then $S_n(F)$, $n \geq 0$, is a hereditary sequence of zigzag languages. Consequently, all of these languages can be generated by Algorithm J.*

In the following we provide simple sufficient conditions guaranteeing that a pattern is tame (see also Remark 15 below).

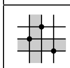
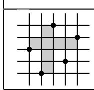
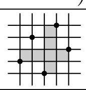
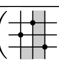
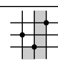
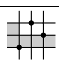
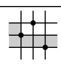
Lemma 9. *If a pattern $\tau \in S_k$, $k \geq 3$, does not have the largest value k at the leftmost or rightmost position, then it is tame.*

We prove Lemma 9 in Section 3.8 below.

Table 1 lists several tame patterns and the combinatorial objects encoded by the corresponding zigzag languages. The bijections between those permutations and the combinatorial objects are well-known and are described in the listed papers (recall also Section 2.8). Some patterns are interesting in their own right and have no 'natural' associated combinatorial objects (empty second column). The resulting ordering for 231-avoiding permutations of length $n = 4$, and the corresponding Gray codes for three different Catalan objects are shown in Figure 3. We refer to the permutation patterns discussed so far as *classical* patterns. In the following we discuss some other important variants of permutation patterns appearing in the literature.

3.3. **Vincular patterns.** Vincular patterns were introduced by Babson and Steingrímsson [BS00]. In a *vincular* pattern τ, there is exactly one underlined pair of consecutive entries, with the interpretation that a match of τ in π requires that the underlined entries match adjacent positions in π. For instance, the permutation $\pi = 3142$ contains the pattern $\tau = 231$, but it avoids the vincular pattern $\tau = \underline{23}1$.

Table 1. Tame permutation patterns and corresponding combinatorial objects and orderings generated by Algorithm J. The patterns in the second part of the table (bottom four rows) are tame after the indicated elementary transformations.

Tame patterns	**Combinatorial objects and ordering**	**References/OEIS** [oei19]
none	permutations by adjacent transpositions → plain change order	[Joh63, Tro62], A000142
$231 = 2\underline{31}$	*Catalan families* • binary trees by rotations → Lucas--van Baronaigien-Ruskey order • triangulations by edge flips • Dyck paths by hill flips	A000108 [LvBR93]
$\underline{23}1$	*Bell families* • set partitions by element exchanges → Kaye's order	A000110 [Kay76, Wil13]
$132 \wedge 231 = \underline{13}2 \wedge 2\underline{31}$: permutations without peaks	binary strings by bitflips → reflected Gray code order (BRGC)	[Gra53], A011782
1342	forests of $\beta(0,1)$-trees	[Bón97, AKPV16], A022558
2143: vexillary permutations		[LS85], A005802
conjunction of v_k tame patterns with $v_2 = 35, v_3 = 91, v_4 = 2346$ (see [Bil13]): k-vexillary permutations ($k \geq 1$)		[BP14], A224318, A223034, A223905
$2143 \wedge 3412$: skew-merged permutations		[Sta94, Atk98], A029759
$2143 \wedge 2413 \wedge 3142$		[DMR10, SV14], A033321
$2143 \wedge 2413 \wedge 3142 \wedge 3412$: X-shaped permutations		[Wat07, Eli11], A006012
$2413 \wedge 3142$: separable permutations	*Schröder families* • slicing floorplans (=guillotine partitions) • topological drawings of $K_{2,n}$	A006318 [AN81, BBL98, ABP06] [CF18]
$2\underline{41}3 \wedge 3\underline{14}2$: Baxter $2\underline{41}3 \wedge 3\underline{41}2$: twisted Baxter $2\underline{14}3 \wedge 3\underline{14}2$	mosaic floorplans (=diagonal rectangulations=R-equivalent rectangulations)	[YCCG03, ABP06] [LR12, CSS18] A001181
$2\underline{14}3 \wedge 3\underline{41}2$	S-equivalent rectangulations	[ABBM$^+$13], A214358
$2\underline{14}3 \wedge 3\underline{41}2 \wedge 2413 \wedge 3142$	S-equivalent guillotine rectangulations	[ABBM$^+$13], A078482
$3\underline{51}24 \wedge 3\underline{51}42 \wedge 24\underline{51}3 \wedge 42\underline{51}3$: 2-clumped permutations	generic rectangulations (=rectangular drawings)	[Rea12b]
conjunction of c_k tame patterns with $c_k = 2(k/2)!(k/2+1)!$ for k even and $c_k = 2((k+1)/2)!^2$ for k odd: k-clumped permutations		[Rea12b]
conjunction of 12 tame patterns: perm. with 0-1 Schubert polynomial		[FMSD19]
$2143 \wedge 2413 \wedge 3412 \wedge 314562 \wedge 412563 \wedge 415632 \wedge 431562 \wedge 512364$ $\wedge\, 512643 \wedge 516432 \wedge 541263 \wedge 541632 \wedge 543162$: widdershins permut.		[BEV18]
	$(\mathbf{2}+\mathbf{2})$-free posets	[Par09, BMCDK10] A022493
$3\overline{1}52\overline{4} = 3\overline{1}42 \wedge 241\overline{3}$		[Pud08, BMCDK10], A098569
$243\overline{1}$ (A051295); $25\overline{3}14$ (A117106), $352\overline{4}1$ (A137534); $4\overline{2}513$ (A137535) $425\overline{1}3$ (A110447); $\overline{4}2153$ (A137536); $253\overline{1}4$ (A137538); $\overline{4}1523$ (A137539) $\overline{4}1253$ (A137540); $3524\overline{1}$ (A137542)		[Pud10]
 $\wedge$  : permutations that characterize Schubert varieties which are Gorenstein		[WY06], A097483
$\mathrm{rot}(2341 \wedge 3\overline{5}241) = 1432 \wedge 1352\overline{4}$: 2-stack sortable permutations	rooted non-separable planar maps	[Wes90, Zei92, GW96] [DGG98], A000139
$2413 \wedge 41\overline{3}52$; $2413 \wedge 45\overline{3}12$; $2413 \wedge 21\overline{3}54$; $3241 \wedge \overline{2}4153$; $3214 \wedge \overline{2}4135$ $\mathrm{rev}(2413 \wedge 41\overline{3}52) = 3142 \wedge 2\underline{41}3$		[DGW96] [CKS09]
conjunction of 20 patterns τ_i with tame $\mathrm{cpl}(\tau_i)$: permutations generated by a stack of depth two and an infinite stack		[Eld06], A245233
$\mathrm{inv}(132 \wedge 312) = 132 \wedge 231$: Gilbreath permutations		[Vel03, DG12]
$\mathrm{rot}(3\overline{1}\underline{42} \wedge 3\overline{1}\underline{24}) = \mathrm{rot}\Big($  $\wedge$  $\Big) =$  $\wedge$  : perms. that uniquely encode pile configurations in patience sorting		[Lan07, BL10], A129698

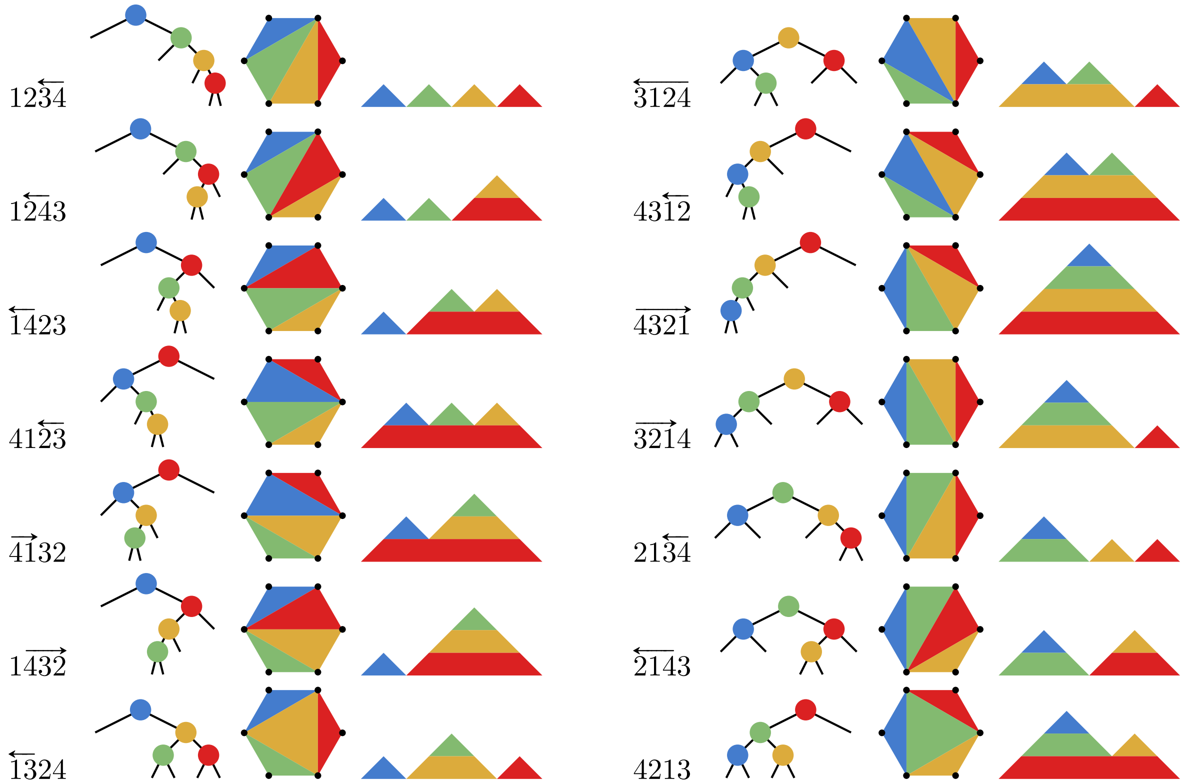


FIGURE 3. 231-avoiding permutations of length $n = 4$ generated by Algorithm J and the resulting Gray codes for Catalan families (binary trees, triangulations, Dyck paths), with jumps indicated by arrows.

$12\overleftarrow{34}$	1|2|3|4	$\overleftarrow{314}2$	13|24
$1\overleftarrow{24}3$	1|2|34	$43\overleftarrow{12}$	134|2
$\overleftarrow{14}23$	1|24|3	$\overrightarrow{4321}$	1234
$41\overleftarrow{23}$	14|2|3	$\overrightarrow{321}4$	123|4
$\overrightarrow{41}32$	14|23	$21\overleftarrow{34}$	12|3|4
$1\overrightarrow{432}$	1|234	$\overleftarrow{214}3$	12|34
$\overleftarrow{13}24$	1|23|4	4213	124|3
$31\overleftarrow{24}$	13|2|4		

FIGURE 4. $\underline{23}1$-avoiding permutations of length $n = 4$ generated by Algorithm J and resulting Gray code for set partitions.

Lemma 10. *If a vincular pattern $\tau \in S_k$, $k \geq 3$, does not have the largest value k at the leftmost or rightmost position, and the largest value k is part of the vincular pair, then it is tame.*

We prove Lemma 10 in Section 3.8 below.

Table 1 also lists several tame vincular patterns and the combinatorial objects encoded by the corresponding zigzag languages, namely set partitions and different kinds of rectangulations. The resulting ordering for $\underline{23}1$-avoiding permutations of length $n = 4$, and the resulting Gray code for set partitions, is shown in Figure 4. The resulting ordering for twisted Baxter permutations of length $n = 4$, and the resulting Gray code for diagonal rectangulations, is shown in Figure 5.

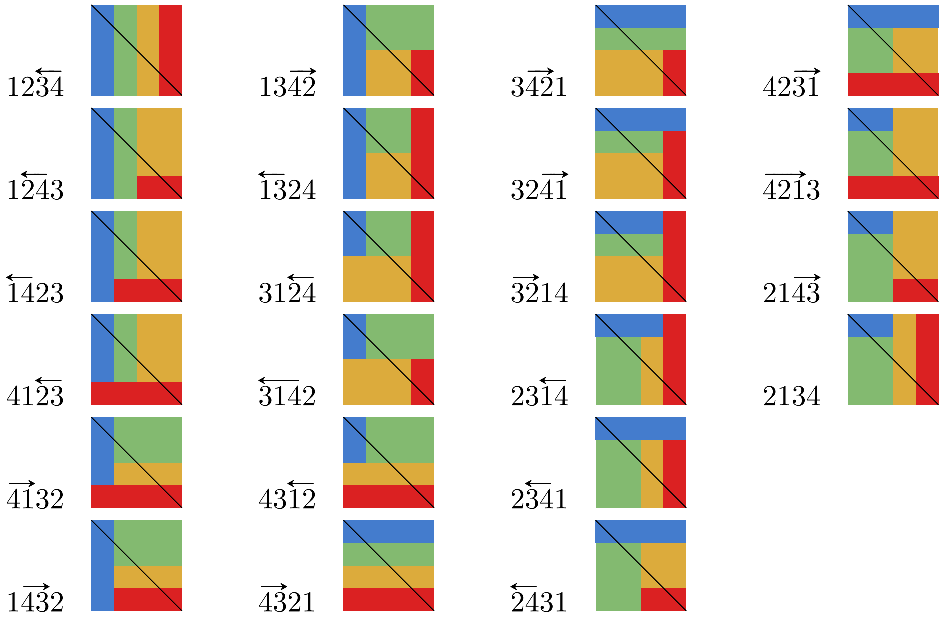


FIGURE 5. Twisted Baxter permutations (2<u>41</u>3 ∧ 3<u>41</u>2-avoiding) for $n = 4$ generated by Algorithm J and resulting Gray code for diagonal rectangulations. Read the figure column by column, from left to right.

3.4. **Barred patterns.** Barred permutation patterns were first considered by West [Wes90]. A *barred* pattern is a pattern τ with a number of overlined entries, e.g., $\tau = 25\bar{3}41$. Let τ' be the permutation obtained by removing the bars in τ, and let τ^- be the permutation that is order-isomorphic to the non-barred entries in τ. In our example, we have $\tau' = 25341$ and $\tau^- = 2431$. A permutation π *contains* a barred pattern τ if and only if it contains a match of τ^- that cannot be extended to a match of τ' by adding entries of π at the positions specified by the barred entries. For instance, $\pi = 35241$ contains $\tau = 25\bar{3}41$, as the highlighted entries form a match of $\tau^- = 2431$ that cannot be extended to a match of $\tau' = 25341$. We clearly have $S_n(\tau^-) \subseteq S_n(\tau)$.

The following lemma gives a sufficient condition for a single-barred pattern to be tame.

Lemma 11. *If for a single-barred pattern $\tau \in S_k$, $k \geq 4$, the permutation $\tau^- \in S_{k-1}$ does not have the largest value $k-1$ at the leftmost or rightmost position, and the barred entry in τ is smaller than k or at a position next to the entry $k-1$, then τ is tame.*

We prove Lemma 11 in Section 3.8 below. See Table 1 for several examples of tame single-barred patterns that were studied by Pudwell [Pud10]. As we will show in Section 4.3 below, in many cases patterns with multiple bars can be reduced to single-barred patterns.

3.5. **Patterns with Bruhat restrictions.** *Patterns with Bruhat restrictions* were introduced by Woo and Yong [WY06]. Such a pattern is a pair (τ, B), where $\tau \in S_k$ and $B \subseteq [k]^2$ is a set of pairs of indices (a, b) with $a < b$ and $\tau(a) < \tau(b)$ such that for all $i \in \{a+1, \ldots, b-1\}$ we either have $\tau(i) < \tau(a)$ or $\tau(i) > \tau(b)$. A permutation π *contains* this pattern if and only if it contains a match of τ, and for any pair of entries $\pi(i_a)$ and $\pi(i_b)$ that are matched by a corresponding pair of entries $\tau(a)$ and $\tau(b)$ with $(a, b) \in B$, we have that $\pi(i) < \pi(i_a)$ or $\pi(i) > \pi(i_b)$ for all $i \in \{i_a + 1, \ldots, i_b - 1\}$.

Lemma 12. *Given a pattern with Bruhat restrictions (τ, B) with $\tau \in S_k$ and $k \geq 3$, if τ does not have the largest value k at the leftmost or rightmost position, then it is tame.*

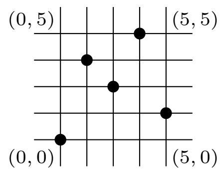
Note that Lemma 12 does not impose any restrictions on the set B, and that it hence generalizes Lemma 9, which corresponds to the case $B = \emptyset$.

3.6. **Bivincular patterns.** *Bivincular patterns* were introduced by Bousquet-Mélou, Claesson, Dukes, and Kitaev [BMCDK10]. Such a pattern is a pair (τ, B), where $\tau \in S_k$ is a vincular pattern and $B \subseteq [k-1]$. A permutation π *contains* this pattern if and only if it contains a match of the vincular pattern τ (respecting the adjacency condition for the vincular pair), and in this match, the entries at positions $\tau^{-1}(i)$ and $\tau^{-1}(i+1)$ are consecutive values in π for all $i \in B$.

Lemma 13. *Given a bivincular pattern (τ, B) with $\tau \in S_k$ and $k \geq 3$, if the vincular pattern τ satisfies the conditions in Lemma* 10 *and if $k-1 \notin B$, then it is tame.*

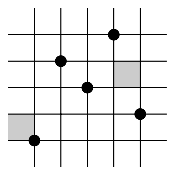
Note that Lemma 13 generalizes Lemma 10, which corresponds to the case $B = \emptyset$. In Table 1, $(\mathbf{2}+\mathbf{2})$-free posets are mentioned as an example of a combinatorial class that is in bijection to permutations avoiding a tame bivincular pattern.

3.7. **Mesh patterns.** We may represent any permutation $\pi \in S_n$ by the set of *points* $(i, \pi(i))$, $1 \leq i \leq n$, in the integer grid $[n]^2$. This *grid representation* is a graphical representation of the permutation matrix. For instance, the permutation $\pi = 14352$ has the grid representation shown on the right. A *cell* in this representation is a connected region in $\mathbb{R}^2 \setminus (\mathbb{R} \times [n] \cup [n] \times \mathbb{R})$, and we number those cells by a pair of integers (a, b), $a, b \in \{0, \ldots, n\}$, from left to right and from bottom to top, as shown in the figure on the right.



Mesh patterns were introduced by Brändén and Claesson [BC11], and they generalize all the aforementioned types of patterns.

A *mesh pattern* is a pair $\sigma = (\tau, C)$, $\tau \in S_k$, with $C \subseteq \{0, \ldots, k\}^2$. Each pair $(a, b) \in C$ encodes a cell numbered (a, b) in the grid representation of τ, and we draw those cells shaded in the grid representation. For instance, the mesh pattern $(\tau, C) = (14352, \{(0,1), (4,3)\})$ has the grid representation shown on the right. These cells from C are the forbidden regions for values of π when searching for a match of $\sigma = (\tau, C)$ in π. Specifically, a permutation π *contains* the mesh pattern σ, if and only if the grid representation of π contains a subset of points that forms the grid representation of τ such that in this match the cells C do not contain any points from π. For example, the permutation 14352 contains the mesh pattern shown on the right, but the permutation 153642 avoids it.



The following main theorem of this section implies all the lemmas about classical, vincular, barred patterns, etc. stated in the previous sections.

Theorem 14. *Let $\sigma = (\tau, C)$, $\tau \in S_k$, $k \geq 3$, be a mesh pattern, and let i be the position of the largest value k in τ. If the pattern satisfies each of the following four conditions, then it is tame:*

(i) i is different from 1 and k.
(ii) For all $a \in \{0, \ldots, k\} \setminus \{i-1, i\}$, we have $(a, k) \notin C$.
(iii) If $(i-1, k) \in C$, then for all $a \in \{0, \ldots, k\} \setminus \{i-1\}$ we have $(a, k-1) \notin C$ and for all $b \in \{0, \ldots, k-2\}$ we have that $(i, b) \in C$ implies $(i-1, b) \in C$.
(iv) If $(i, k) \in C$, then for all $a \in \{0, \ldots, k\} \setminus \{i\}$ we have $(a, k-1) \notin C$ and for all $b \in \{0, \ldots, k-2\}$ we have that $(i-1, b) \in C$ implies $(i, b) \in C$.

The conditions in Theorem 14 can be understood in the grid representation of (τ, C) as follows; see the left hand side of Figure 6: Condition (i) asserts that the highest point of τ must not be the leftmost or rightmost point (the two crossed out grid points in the figure are forbidden).

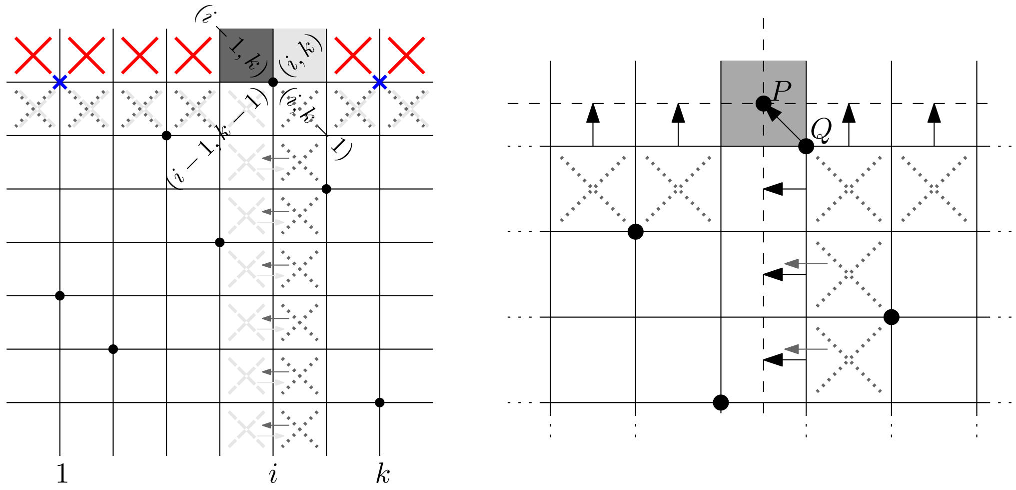


Figure 6. Illustration of the four conditions in Theorem 14 (left) and how they are used in the proof of the theorem (right).

Condition (ii) asserts that none of the cells in the topmost row (above the points) must be shaded, with the possible exception of the cells next to the highest point (solid crossed out cells in the figure). Condition (iii) asserts that if the cell $(i-1,k)$ to the top left of the highest point is shaded (dark gray cell in the figure), then none of the cells in the row below except possibly $(i-1,k-1)$ must be shaded (dotted crossed out cells without arrows in the figure), and if one of the cells strictly below $(i,k-1)$ is shaded, then the cell to the left of it must also be shaded (dotted crossed out cells with arrows). Symmetrically, condition (iv) asserts that if the cell (i,k) to the top right of the highest point is shaded (light gray cell in the figure), then none of the cells in the row below except possibly $(i,k-1)$ must be shaded (dashed crossed out cells without arrows in the figure), and if one of the cells strictly below $(i-1,k-1)$ is shaded, then the cell to the right of it must also be shaded (dashed crossed out cells with arrows).

Proof. We show that if $\sigma=(\tau,C)$ satisfies the four conditions of the theorem, then $S_n(\sigma)$, $n\geq 0$, is a hereditary sequence of zigzag languages. We argue by induction on n. Note that $S_0(\sigma)=S_0=\{\varepsilon\}$ is a zigzag language by definition, so the induction basis is clear. For the induction step let $n\geq 1$. We first show that if $\pi\in S_{n-1}(\sigma)$, then $c_1(\pi),c_n(\pi)\in S_n(\sigma)$. As $c_1(\pi)$ and $c_n(\pi)$ are obtained from π by inserting the new largest value n at the leftmost or rightmost position, respectively, the grid representation of these two permutations differs from the grid representation of π by adding a new highest point at the leftmost or rightmost position. However, as π avoids σ by assumption, condition (i) guarantees that both $c_1(\pi)$ and $c_n(\pi)$ also avoid σ, which is what we wanted to show.

To complete the induction step, we now show that if $\pi\in S_n(\sigma)$, then $p(\pi)\in S_{n-1}(\sigma)$. Recall that $p(\pi)$ is obtained from π by removing the largest value n, so in the grid representation, we remove the highest point P. Our assumption is that π avoids the pattern σ, and we need to show that removing the highest point does not create a match of the pattern σ. For the sake of contradiction, suppose that removing P creates a match of the pattern σ in $p(\pi)$. Let Q be the highest point in this match of the pattern σ in $p(\pi)$. This situation is illustrated on the right hand side of Figure 6. By condition (ii), we are in exactly one of the following two cases: (a) the cell $(i-1,k)$ is in C and P lies inside this cell of σ in this match of the pattern; (b) the cell (i,k) is in C and P lies inside this cell of σ in this match of the pattern. We first consider case (a): We claim that we can exchange the point Q for the point P in the match of the pattern σ, and obtain another match of σ in π, which would contradict the assumption that π avoids σ. Indeed, this exchange operation strictly enlarges only the cells $(a, k-1)$ for all $a \in \{0, \ldots, k\} \setminus \{i-1\}$ and the cells (i, b) for all $b \in \{0, \ldots, k-2\}$. The first set of cells are not in C by the first part of condition (iii). The second set of cells are either not in C, or if they are, then the corresponding cells to the left of it are also in C by the second part of condition (iii). Moreover, after the exchange the cell (i, k) contains no point from π, as P is the highest point (this is of course only relevant if $(i, k) \in C$). Furthermore, after the exchange the cell $(i-1, k-1)$ contains at most those points from π that were in the same cell before the exchange (clearly P is the only point inside the cell $(i-1, k)$). So we indeed obtain a match of σ in π, a contradiction.

In the symmetric case (b), we apply the same exchange argument, using condition (iv) instead of (iii). This completes the proof. □

3.8. **Proof of Lemmas 9–13.** With Theorem 14 in hand, the proofs of Lemmas 9–13 are straightforward. As noted before, Lemma 12 generalizes Lemma 9, and Lemma 13 generalizes Lemma 10, so we only need to prove Lemmas 11, 12 and 13.

Proof of Lemma 11. Note that a barred pattern $\tau \in S_k$ with a single barred entry b at position a corresponds to the mesh pattern $\sigma = (\tau^-, \{(a-1, b-1)\})$, i.e., in the grid representation of σ a single cell is shaded. It follows that conditions (iii) and (iv) of Theorem 14 are trivially satisfied, and conditions (i) and (ii) translate into the conditions in the lemma. □

Proof of Lemma 12. A pattern with Bruhat restrictions (τ, B) corresponds to the mesh pattern $\sigma = (\tau, C)$ where C is the union of all the sets $R(a, b) := \{(i, j) \mid a \leq i < b \wedge \tau(a) \leq j < \tau(b)\}$ for $(a, b) \in B$, i.e., in the grid representation of σ, certain rectangles of cells inside the bounding box of the points from τ are shaded. It follows that conditions (ii)–(iv) of Theorem 14 are trivially satisfied, and condition (i) corresponds exactly to the condition in the lemma. □

Proof of Lemma 13. A vincular pattern $\tau \in S_k$ where the entries at positions a and $a+1$ are underlined corresponds to the mesh pattern $\sigma = (\tau, C)$ with $C := \{a\} \times \{0, \ldots, k\}$, i.e., in the grid representation of σ, an entire column of cells is shaded. For the bivincular pattern (τ, B) we also have to add the sets $\{0, \ldots, k\} \times \{b\}$ for all $b \in B$ to the set of cells C, i.e., in the grid representation we also have to shade the corresponding rows of cells. By the conditions stated in Lemma 10, conditions (i) and (ii) of Theorem 14 are satisfied. By the condition $k-1 \notin B$, conditions (iii) and (iv) of the theorem are also satisfied, proving that the mesh pattern σ is tame. □

Remark 15. One can argue that if condition (i) in Theorem 14 is violated, then $S_k(\sigma)$ is not a zigzag language. Similarly, if condition (ii) is violated, then $S_k(\sigma) \neq p(S_{k+1}(\sigma))$, i.e., the hereditary property is violated. It follows from the proofs of Lemmas 9–13 before that the conditions stated in those lemmas are not only sufficient, but also necessary for tameness.

3.9. **Patterns with multiplicities.** All the aforementioned notions and results in this section generalize straightforwardly to bounding the number of appearances of a pattern. Formally, a *counted pattern* is a pair $\sigma = (\tau, c)$, where τ is a mesh pattern, and c is a non-negative integer. Moreover, $S_n(\sigma)$ denotes the set of all permutations from S_n that contain *at most* c matches of the pattern τ, where the special case $c = 0$ is pattern-avoidance (cf. [NZ96]).

By Theorem 8, we can now form propositional formulas F consisting of logical ANDs $\wedge$, ORs $\vee$, and tame counted patterns (τ_i, c_i) as variables, with possibly different counts c_i for each variable. The tameness of each (τ_i, c_i) can be checked by verifying whether the patterns τ_i

satisfy the conditions stated in Theorem 14 or its corollaries Lemmas 9–13. We then obtain a hereditary zigzag language $S_n(F)$ that can be generated by Algorithm J.

A somewhat contrived example for such a language would be $F = \big((231,3) \wedge (2143,5)\big) \vee (3\underline{14}2,2)$, the language of permutations that contain at most 3 matches of the pattern 231 AND at most 5 matches of the pattern 2143, OR at most 2 matches of the vincular pattern $3\underline{14}2$.

4. Algebra with patterns

In this section we significantly extend the methods described in the previous section, by applying geometric transformations to permutation patterns, and by describing some other types of patterns as conjunctions and disjunctions of suitable mesh patterns (recall Theorem 8). Two particularly relevant additional types of permutations covered in this section are monotone and geometric grid classes; see Theorem 21 below.

4.1. Elementary transformations.

We now consider three important *elementary transformations* of permutations that are important in the context of pattern-avoidance, as they preserve the cardinality of the set $S_n(F)$. Each of them corresponds to a geometric transformation of the grid representation of each of the patterns $\tau = a_1 \dots a_k$ in the formula F, and together these transformations form the dihedral group D_4 of symmetries of a regular 4-gon:

- *Reversal*, defined as $\mathrm{rev}(\tau) := a_k \dots a_1$. This corresponds to a vertical reflection of the grid representation.
- *Complementation*, defined as $\mathrm{cpl}(\tau)_i = k+1-a_i$ for all $i = 1,\dots,k$. This corresponds to a horizontal reflection of the grid representation.
- *Inversion*, defined by $\mathrm{inv}(\tau)_{\tau(i)} = i$ for all $i = 1,\dots,k$. This corresponds to a diagonal reflection of the grid representation along the south-west to north-east diagonal.

Note that a clockwise 90-degree rotation is obtained as $\mathrm{rot}(\tau) := \mathrm{inv}(\mathrm{rev}(\tau)) = \mathrm{cpl}(\mathrm{inv}(\tau))$. Clearly, all these operations generalize to mesh patterns (τ, C), by applying the aforementioned geometric transformations to the cells in C. These operations and their relations are illustrated in Figure 7 for $(\tau, C) = (14352, \{(1,0),(1,1),(3,3),(4,3)\})$.

The following lemma is immediate.

Lemma 16. *Given any composition h of the elementary transformations reversal, complementation and inversion, and any propositional formula F consisting of logical ANDs $\wedge$, ORs $\vee$, and mesh patterns $\tau_1,\dots,\tau_\ell$ as variables, then the sets of pattern-avoiding permutations $S_n(F)$ and $S_n(h(F))$ are in bijection under h for all $n \geq 1$, where the formula $h(F)$ is obtained from F by replacing every pattern τ_i by $h(\tau_i)$ for all $i = 1,\dots,\ell$.*

Lemma 16 is very useful for the purpose of exhaustive generation, because even if τ_i is not tame, then maybe $h(\tau_i)$ is. So even if we cannot apply Algorithm J to generate $S_n(\tau)$ directly, we may be able to generate $S_n(h(\tau_i))$, and then apply h^{-1} to the resulting permutations. For instance, $\tau = 213$ is not tame, as the largest entry appears at the rightmost position. However, $\mathrm{cpl}(\tau) = 231$ is tame by Lemma 9, and so we can use Algorithm J to generate $S_n(\mathrm{cpl}(\tau))$.

As another example, consider so-called *2-stack sortable permutations* introduced by West [Wes90] and later counted in [Zei92, GW96, DGG98]. These permutations are characterized by the pattern-avoidance formula $F = \tau_1 \wedge \tau_2$ with $\tau_1 := 2341$ and $\tau_2 := 3\overline{5}241$ (τ_2 is a barred pattern). Unfortunately, τ_2 is not tame (the barred entry $\overline{5}$ is not at a position next to the entry 4; recall Lemma 11), so Algorithm J cannot be used directly for generating $S_n(F)$. However, applying rotation, $h(\tau) := \mathrm{rot}(\tau) = \mathrm{inv}(\mathrm{rev}(\tau))$, yields two tame patterns $h(\tau_1) = 1432$ and $h(\tau_2) = 1352\overline{4}$ and the formula $h(F) = h(\tau_1) \wedge h(\tau_2)$, which can be used for generating $S_n(h(F))$ via Algorithm J:

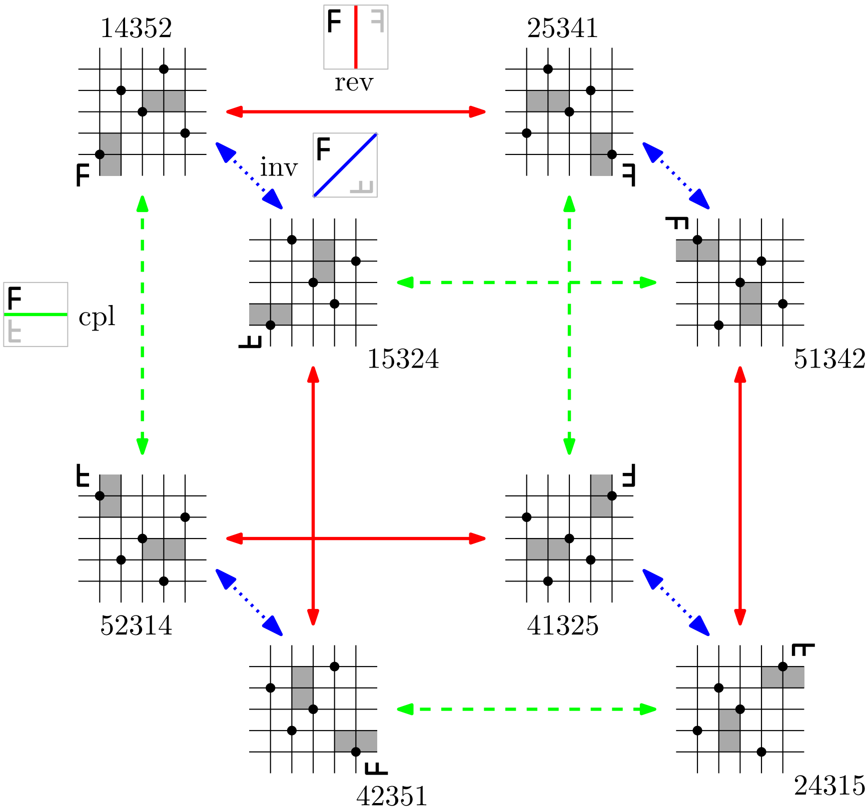
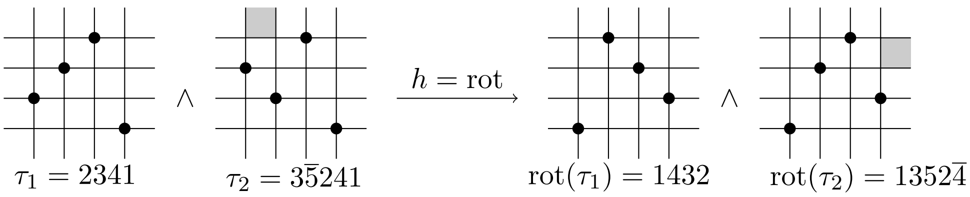
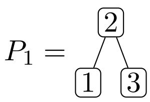


FIGURE 7. Elementary transformations between permutations.



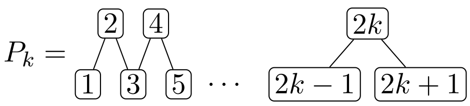
The bottom part of Table 1 lists further pattern-avoiding permutations that have been studied in the literature and that can be turned into tame patterns by elementary transformations.

4.2. **Partially ordered patterns.** Partially ordered patterns were introduced by Kitaev [Kit05]. A *partially ordered pattern (POP)* is a partially ordered set $P = ([k], \prec)$, and we say that a permutation π *contains* this pattern if and only if it contains a subpermutation $a_{i_1} \ldots a_{i_k}$, $i_1 < \cdots < i_k$, such that $k \prec l$ in the partial order implies that $a_{i_k} < a_{i_l}$. In particular, if $\prec$ is a linear order, then this is equivalent to classical pattern avoidance. However, some other constraints can be expressed much more conveniently using POPs. For instance, avoiding the POP



is equivalent to avoiding peaks in the permutation, so $S_n(P_1)$ is the set of permutations without peaks discussed before, which satisfies $|S_n(P_1)| = 2^{n-1}$.

More generally, the POP



realizes the language $S_n(P_k)$ of permutations with at most $k-1$ peaks.

We let $L(P)$ denote the set of all linear extensions of the poset P, and for any linear extension $x \in L(P)$, we consider the inverse permutation of x, as the ith entry of $\mathrm{inv}(x)$ denotes the position of i in x. Moreover, $\mathrm{inv}(x) \in S_k$, so $\mathrm{inv}(x)$ is a classical pattern.

Lemma 17. *For any partially ordered pattern $P = ([k], \prec)$, we have*

$$S_n(P) = \bigcap_{x \in L(P)} S_n(\mathrm{inv}(x)) = S_n\Big(\bigwedge\nolimits_{x \in L(P)} \mathrm{inv}(x)\Big). \tag{3}$$

In particular, if the poset P does not have 1 or k as a maximal element, then P is tame.

Proof. The first part of the lemma follows immediately from the definition of POPs and from (2). To prove the second part, suppose that P does not have 1 or k as a maximal element. Then in any linear extension $x \in L(P)$, 1 and k will not appear at the last position, and so in the inverse permutation $\mathrm{inv}(x)$, the largest entry k will neither be at position 1 nor at position k. We can hence apply Lemma 9, and using Theorem 8 we obtain that P is tame. □

For instance, for the POP P_1 from before we have $L(P_1) = \{132, 312\}$, and so $P_1 = 132 \wedge 231$, and for the POP P_2 we have $L(P_2) = \{13254, 13524, 13542, 15324, 15342, 31254, \ldots\}$, a set of 16 linear extensions in total, so $P_2 = 13254 \wedge 14253 \wedge 15243 \wedge 14352 \wedge 15342 \wedge 23145 \wedge \cdots$.

Moreover, we can create counted POPs with multiplicity c (recall Section 3.9), by taking the OR of conjunctions of counted classical patterns as described by Lemma 17, over all number partitions of c into the corresponding number of parts. For instance, the counted POP $\sigma = (P_1, c)$, which realizes the zigzag language $S_n(\sigma)$ of permutations with at most c triples of values forming a peak, is obtained by considering the partitions $c = c+0 = (c-1)+1 = \cdots = 1+(c-1) = 0+c$, resulting in the formula

$$(P_1, c) = \big((132, c) \wedge (231, 0)\big) \vee \big((132, c-1) \wedge (231, 1)\big) \vee \cdots \vee \big((132, 0) \wedge (231, c)\big)$$

with counted classical patterns on the right-hand side.

4.3. **Barred patterns with multiple bars.** Patterns with multiple bars can be reduced to single-barred patterns (to which Lemma 11 applies) as shown by the following lemma.

Lemma 18 (cf. [Úlf11]). *Let $\tau \in S_k$, $k \geq 5$, be a pattern with $b \geq 2$ bars, such that no two barred entries are at neighboring positions or have adjacent values. Let $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_b \in S_{k-b+1}$ be the permutations with a single barred entry that are order-isomorphic to the sequences obtained from τ by removing all but except one barred entry. Then we have*

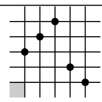
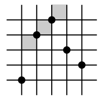
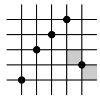
$$S_n(\tau) = \bigcap_{1 \leq i \leq b} S_n(\widetilde{\tau}_i) = S_n\Big(\bigwedge\nolimits_{1 \leq i \leq b} \widetilde{\tau}_i\Big).$$

Consequently, if $\tau^- \in S_{k-b}$ does not have the largest value $k-b$ at the leftmost or rightmost position, and the largest barred entry in τ is smaller than k or at a position next to the entry $k-1$, than τ is tame.

Proof. To prove the first part, observe that when no two barred entries are at neighboring positions or have adjacent values, then the definition of barred pattern avoidance is equivalent to avoiding each of the single-barred patterns $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_b$, so the claim follows using (2).

To prove the second part we show that each of the single-barred patterns $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_b$ satisfies the conditions of Lemma 11. Indeed, we know that $\tau^- = (\widetilde{\tau}_i)^- \in S_{k-b}$, $1 \leq i \leq b$, does not have the largest value $k-b$ at the leftmost or rightmost position. Moreover, if $\widetilde{\tau}_i$ is obtained from τ by removing all but the largest barred entry, then the barred entry in $\widetilde{\tau}_i \in S_{k-b+1}$ is either

Table 2. Illustration of Lemma 19.

τ	τ'	τ^-	r	s	σ
$\overline{1}45632$	145632	34521	1	1	
$1\overline{4}5632$	145632	14532	2	4	
$14\overline{5}632$	145632	14532	2	4	
$145\overline{6}32$	145632	14532	2	4	
$1456\overline{3}2$	145632	13452	5	6	
$14563\overline{2}$	145632	13452	5	6	

smaller than $k-b+1$ or at a position next to the entry $k-b$. To see this note that if the largest entry k in τ is barred, then the second largest entry $k-1$ is not barred by the assumption that no two barred entries have adjacent values. For the same reason, if $\widetilde{\tau}_i$ is obtained from τ by removing barred entries including the largest one, then the barred entry in $\widetilde{\tau}_i \in S_{k-b+1}$ is smaller than $k-b+1$. Consequently, we can apply Lemma 11 to each of the patterns $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_b$, and complete the proof by applying Theorem 8. □

Lemma 18 applies for instance to the tame pattern $3\overline{1}52\overline{4} = 3\overline{1}42 \wedge 241\overline{3}$ listed in Table 1.

4.4. **Weak avoidance of barred patterns and dotted patterns.** Weak pattern avoidance and dotted patterns were introduced by Baril [Bar11] (see also [DRT14]). Given a single-barred pattern τ, we define τ' and τ^- as in Section 3.4 before, and we say that a permutation π *weakly contains* τ, if and only if it contains a match of τ^- that cannot be extended to a match of τ' by adding one entry of π, not necessarily at the position specified by the barred entry. Otherwise, we say that π *weakly avoids* τ. We let $S_n^w(\tau)$ denote the set of permutations that weakly avoid τ. For instance, $\pi = 1243$ contains the barred pattern $\tau = 12\overline{3}$, as for the increasing pair 24 in π, there is no entry in π to the right that extends this pair to an increasing triple. However, π weakly avoids τ, as each of the increasing pairs 12, 14, 13, 24 and 23 can be extended to an increasing triple, by adding an entry from π to the right, middle, middle, left and left of this pair, respectively.

The relation between weak pattern avoidance and mesh pattern avoidance is captured by the following lemma.

Lemma 19. *Let $\tau \in S_k$ be a single-barred pattern with barred entry b. Consider the longest increasing or decreasing substring of consecutive values in τ' that contains b, and let r and s be the start and end indices of this substring. Let σ be the mesh pattern defined by $\sigma := (\tau^-, C)$ with*

$$C := \{(r+i-1, \tau'(r+i)-1) \mid 0 \le i \le s-r\}. \tag{4}$$

Then we have $S_n^w(\tau) = S_n(\sigma)$.

Table 2 illustrates the definition of the mesh pattern $\sigma = (\tau^-, C)$ defined in Lemma 19.

Proof. We only consider the case that the longest substring of consecutive values in τ' that contains b is increasing, as the other case is symmetric.

We first show that if a permutation π weakly contains τ, then it also contains σ. For this consider a match of τ^- that cannot be extended to a match of τ' in the grid representation of π; see the left hand side of Figure 8. Consider the $s-r$ points $P_r, \ldots, P_{s-1}$ of π to which the entries at positions $r, \ldots, s-1$ of τ^- are matched. We know that they form an increasing sequence, and

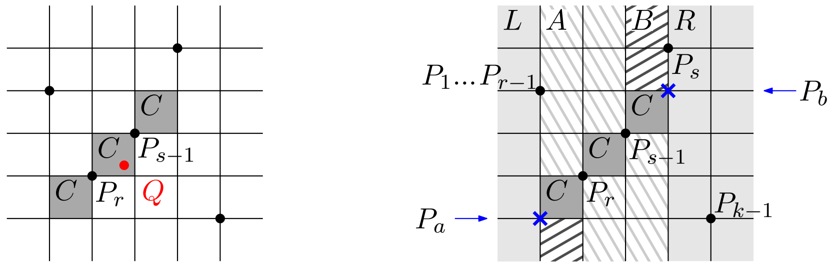


FIGURE 8. Illustration of the proof of Lemma 19.

all other points in this match are below or above them. Now consider the $s-r+1$ cells in C defined in (4) between and around these points. We need to show that none of them contains any points of π, demonstrating that this is a match of the mesh pattern σ. Indeed, if one of these regions did contain a point Q from π, then Q together with $P_r, \dots, P_{s-1}$ would form an increasing sequence of length $s-r+1$, i.e., Q would extend the match of τ^- to a match of τ' in π, a contradiction.

It remains to show that if a permutation π contains the mesh pattern σ, then it weakly contains τ. For this consider a match of $\sigma = (\tau^-, C)$ in the grid representation of π; see the right hand side of Figure 8. We label the points to which the entries of τ^- are matched by $P_1, \dots, P_{k-1}$. By the definition of σ, the points $P_r, \dots, P_{s-1}$ form a longest increasing sequence of consecutive values in this match. In particular, P_{r-1} is not located at the bottom left corner of the leftmost cell of C, and P_s is not located at the top right corner of the rightmost cell of C, so there is a point P_a, $a \in [k-1] \setminus \{r-1, \dots, s-1\}$, on the same height as the lower boundary of the leftmost cell of C, and a point P_b, $b \in [k-1] \setminus \{r, \dots, s\}$, on the same height as the upper boundary of the rightmost cell of C. We know that no point of π lies within any of the cells in C, but we also need to show that a point Q of π contained in any of the other cells cannot be used to extend this match of τ^- to to a match of τ'. For this we distinguish four cases: Suppose that Q is contained in a cell L to the left of the cells in C. Then $P_1, \dots, P_{k-1}$ together with Q is not a match of τ', as the longest increasing substring of consecutive values $P_r, \dots, P_{s-1}$ contains only $s-r$ points. A symmetric argument works if Q is contained in a cell R to the right of the cells in C. Now suppose that Q is contained in a cell A above a cell from C, but not above the rightmost one, or below a cell from C, but not below the leftmost one. In this case the points $P_r, \dots, P_{s-1}$ together with Q do not form an increasing substring, so this is not a match of τ'. It remains to consider the case that Q is contained in a cell B above the rightmost cell from C, or below the leftmost cell from C. In this case the points $P_r, \dots, P_{s-1}$ together with Q form an increasing substring, but the values are not consecutive, as Q is separated from $P_r, \dots, P_{s-1}$ by the point P_b or P_a, respectively, so this is not match of τ' either. This completes the proof. □

Baril [Bar11] also introduced the notion of a *dotted* pattern $\tau \in S_k$, which is a pattern with some entries at positions $I \subseteq [k]$ that have a dot above them. A permutation π *avoids* the pattern τ if and only if π weakly avoids every single-barred pattern obtained by putting a bar above every entry not in I, i.e., we have

$$S_n(\tau) := \bigcap_{j \in [k] \setminus I} S_n^w(\tau_j), \qquad (5)$$

where τ_j is the barred pattern obtained by putting the bar above entry j.

Lemma 20. *Let $\tau \in S_k$, $k \geq 3$, be a dotted pattern with dots over all positions $I \subseteq [k]$. If τ does not have the largest value k at the leftmost or rightmost position, and all entries in the longest increasing or decreasing substring of consecutive values including k are dotted, then τ is tame.*

Proof. Let r and s be the start and end indices of the longest increasing or decreasing substring of consecutive values including k. The conditions of the lemma imply that $[r,s] \subseteq I$.

By (5), Theorem 8, and Lemma 19, to prove that τ is tame it is sufficient to show this for the mesh pattern $\sigma = (\rho^-, C)$ with C defined in (4) for every single-barred pattern ρ obtained from τ' by placing a bar over an entry at a position $[k] \setminus I$. As $[r,s] \subseteq I$, we obtain in particular that the largest value k in ρ is not barred, and if the second largest entry $k-1$ is next to k, then it is also not barred. Combining this with the assumption that the largest value k is not at the leftmost or rightmost position in ρ, we obtain that in ρ^-, the largest entry $k-1$ is not at the leftmost or rightmost position. Moreover, we obtain from the definition (4) that C does not have any cells in the topmost row, i.e., $(i, k-1) \notin C$ for $i = 0, \ldots, k-1$. Therefore, applying Theorem 14 shows that σ is tame, completing the proof. □

4.5. **Monotone and geometric grid classes.** Monotone grid classes of permutations were introduced by Huczynska and Vatter [HV06]. To define them, we consider a matrix M with entries from $\{0,+1,-1\}$, indexed first by columns from left to right, and then by rows from bottom to top. A permutation π of $[n]$ (for any $n \geq 0$) is in the *monotone grid class of* M, denoted $\mathrm{Grid}(M)$, if we can place the points labelled from 1 to n from bottom to top into a rectangular grid that has as many rows and columns as the matrix M, and reading the labels from left to right will yield π, subject to the following conditions; see the top part of Figure 9: No two points are placed on the same horizontal or vertical line. Moreover, for each cell (x,y) in the grid, if $M_{x,y} = 0$ then the cell contains no points, if $M_{x,y} = +1$, then the points in this cell are increasing, and if $M_{x,y} = -1$ then the points in this cell are decreasing. Based on this, we define $\mathrm{Grid}_n(M) := \mathrm{Grid}(M) \cap S_n$. It is an open problem whether $\mathrm{Grid}(M)$ is characterized by finitely many forbidden patterns for any M (cf. [AB16]).

The second type of permutations we shall discuss in this section are geometric grid classes, introduced by Albert, Atkinson, Bouvel, Ruškuc, and Vatter [AAB+13]. They are defined using a matrix M with entries from $\{0,+1,-1\}$ as before. A permutation π of $[n]$ is in the *geometric grid class of* M, denoted $\mathrm{Geo}(M)$, if it can be drawn in a rectangular grid as described before, with the slightly strengthened conditions that if $M_{x,y} = +1$, then the points in the cell (x,y) lie on the increasing diagonal line through this cell, and if $M_{x,y} = -1$ then the points in the cell (x,y) lie on the decreasing diagonal line through this cell; see the bottom part of Figure 9. Similarly to before, we define $\mathrm{Geo}_n(M) := \mathrm{Geo}(M) \cap S_n$. We clearly have $\mathrm{Geo}(M) \subseteq \mathrm{Grid}(M)$ and $\mathrm{Geo}_n(M) \subseteq \mathrm{Grid}_n(M)$.

Unlike for monotone grid classes, it was shown in [AAB+13] that any geometric grid class $\mathrm{Geo}(M)$ is characterized by finitely many forbidden patterns, i.e.,

$$\mathrm{Geo}_n(M) = S_n(\tau_1 \wedge \cdots \wedge \tau_\ell)$$

for a suitable set of patterns $\tau_1, \ldots, \tau_\ell$ and for all $n \geq 0$. For instance, X-shaped permutations studied in [Wat07, Eli11] are exactly the permutations in $S_n(2143 \wedge 2413 \wedge 3142 \wedge 3412)$. However, the argument given in [AAB+13] for the existence of $\tau_1, \ldots, \tau_\ell$ is non-constructive, so there is no procedure known to compute these patterns from the matrix M.

Nevertheless, our next theorem provides an easily verifiable sufficient condition for deciding whether $\mathrm{Grid}_n(M)$ and $\mathrm{Geo}_n(M)$ are zigzag languages, based only on two particular entries of M.

Theorem 21. *Let M be a matrix with -1 in the top-left corner and $+1$ in the top-right corner. Then $\mathrm{Grid}_n(M)$, $n \geq 0$, and $\mathrm{Geo}_n(M)$, $n \geq 0$, are both hereditary sequences of zigzag languages. Consequently, all of these languages can be generated by Algorithm J.*

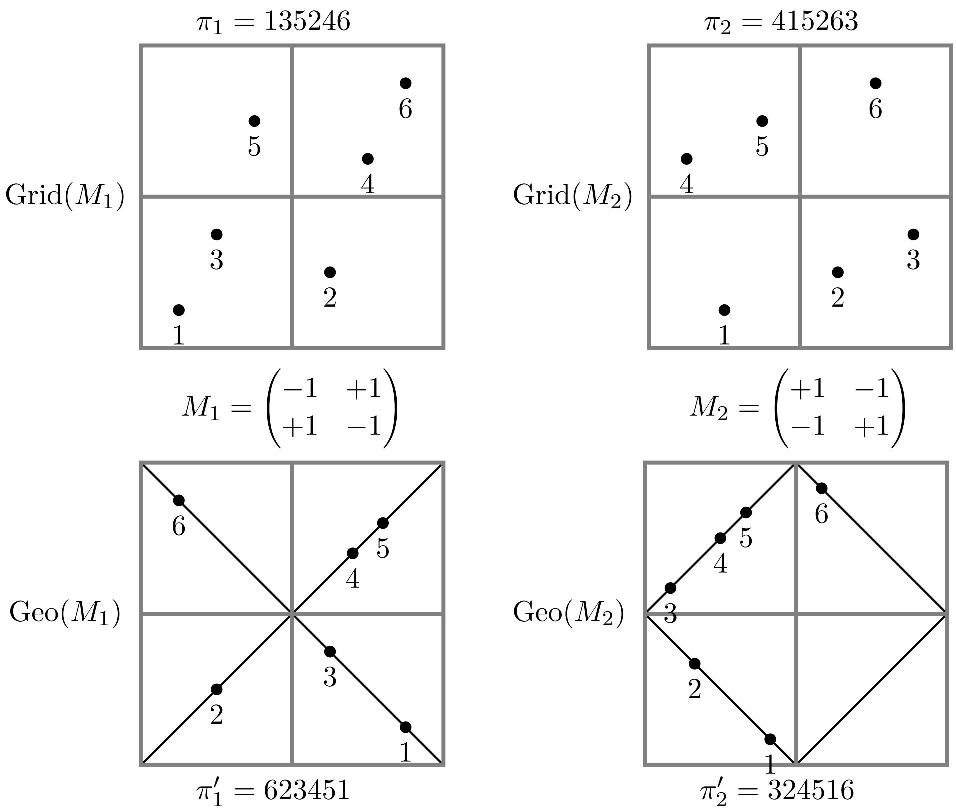


Figure 9. Top: Illustration of monotone grid classes. Bottom: Illustration of geometric grid classes, with X-shaped permutations on the left, and O-shaped permutations on the right. Observe that $\pi_1 \in \mathrm{Grid}(M_1)$, but $\pi_1 \notin \mathrm{Geo}(M_1)$, as π_1 contains the pattern 2413. Similarly, we have $\pi_2 \in \mathrm{Grid}(M_2)$, but $\pi_2 \notin \mathrm{Geo}(M_2)$, as π_2 contains the pattern 1423.

From the two monotone and geometric grid classes shown in Figure 9, only the left two satisfy the conditions of the theorem.

Proof. We only prove the theorem for monotone grid classes $\mathrm{Grid}_n(M)$. The argument for geometric grid classes $\mathrm{Geo}_n(M)$ is completely analogous.

We argue by induction on n. Note that $\mathrm{Grid}_0(M) = S_0 = \{\varepsilon\}$ is a zigzag language by definition, so the induction basis is clear. For the induction step let $n \geq 1$. We first show that if $\pi \in \mathrm{Grid}_{n-1}(M)$, then $c_1(\pi), c_n(\pi) \in \mathrm{Grid}_n(M)$. For this argument we use the assumption that the top-left entry of M is -1, and the top-right entry of M is $+1$, i.e., π can be drawn into a grid so that the points in the top-left cell are decreasing, and the points in the top-right cell are increasing. It follows that we can draw $c_1(\pi)$ on the same grid, by extending the drawing of π so that the new point n is placed to the top-left of all other points. Similarly, we can draw $c_n(\pi)$ on the same grid, by extending the drawing of π so that the new point n is placed to the top-right of all other points.

To complete the induction step, we now show that if $\pi \in \mathrm{Grid}_n(M)$, then $p(\pi) \in \mathrm{Grid}_{n-1}(M)$. As $\pi \in \mathrm{Grid}_n(M)$, we can draw π into a grid respecting the monotonicity conditions described by M. Clearly, removing any entry from π, in particular the largest one, maintains this property, i.e., we can draw $p(\pi)$ on the same grid, showing that $p(\pi) \in \mathrm{Grid}_{n-1}(M)$. This completes the proof. □

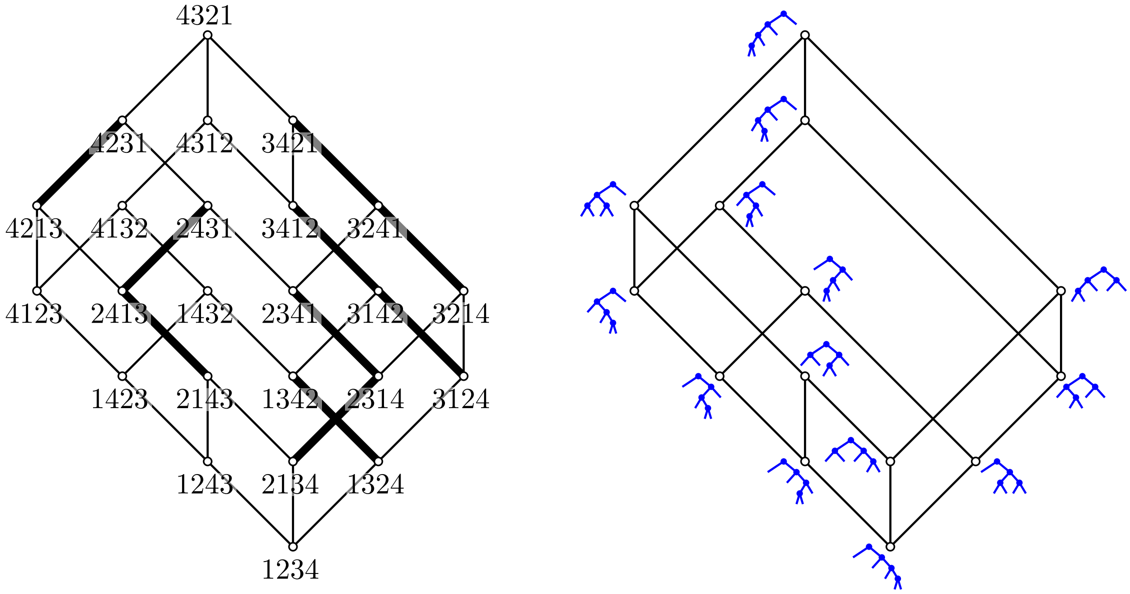


FIGURE 10. Hasse diagrams of the weak order on S_4 (left), with the lattice congruence for 231-avoiding permutations (bold edges), and of the resulting lattice quotient $S_n/\equiv$ (right), which is the well-known Tamari lattice (with corresponding binary trees).

5. Lattice congruences of the weak order

The second main application of our framework are lattice congruences of the weak order on the symmetric group S_n. With lattice congruences, we obtain a provable notion of optimality for the Gray codes obtained from Algorithm J, in the sense that jump operations translate into walks along the edges of a polytope. The main results in this section are summarized in Theorem 22 and Corollary 23. Proofs of these results will be presented in part II of this paper series.

5.1. **Preliminaries.** We begin recalling a few basic notion from poset theory. A *partially ordered set*, or *poset* for short, is a pair $(P,<)$, where P is a set and $<$ is a reflexive, antisymmetric and transitive binary relation on P. A *cover relation* is a pair $x,y\in P$ with $x<y$ for which there is no $z\in P$ with $x<z<y$. In this case we say that y *covers* x and we write $x\lessdot y$. Clearly, the cover relations form an acyclic directed graph with vertex set P, and this graph is referred to as the *cover graph of* P. A drawing of the cover graph with all cover edges $x\lessdot y$ leading upwards is called a *Hasse diagram*. A poset $(P,<)$ is called a *lattice*, if for any two $x,y\in P$ there is a unique smallest element z, called the *join* $x\vee y$ *of* x *and* y, such that $z>x$ and $z>y$, and if there is unique largest element z, called the *meet* $x\wedge y$ *of* x *and* y, satisfying $z<x$ and $z<y$. A *lattice congruence* is an equivalence relation $\equiv$ on P such that $x\equiv x'$ and $y\equiv y'$ implies that $x\vee y\equiv x'\vee y'$ and $x\wedge y\equiv x'\wedge y'$. Given any lattice congruence $\equiv$, we obtain the *lattice quotient* $P/\equiv$ (which is itself a lattice) by taking the equivalence classes as elements, and ordering them by $X<Y$ if and only if there is an $x\in X$ and a $y\in Y$ such that $x<y$ in P. Observe that the cover graph of $P/\equiv$ is obtained from the cover graph of P by contracting all cover edges $x\lessdot y$ with $x\equiv y$.

The *weak order* on the symmetric group S_n is obtained by considering the *inversion set* of a permutation, defined as

$$\mathrm{inv}(\pi):=\{(\pi(i),\pi(j))\mid 1\leq i<j\leq n \text{ and } \pi(i)>\pi(j)\},$$

and by defining $\pi < \rho$ if and only if $\mathrm{inv}(\pi) \subseteq \mathrm{inv}(\rho)$; see the left hand side of Figure 10. The cover relations $\pi \lessdot \rho$ in this poset are exactly adjacent transpositions. The weak order forms a lattice, where the inversion set of the join $\pi \vee \rho$ of two permutations π and ρ is given by the transitive closure of $\mathrm{inv}(\pi) \cup \mathrm{inv}(\rho)$, and the inversion set of the meet can be computed similarly by considering the reverse permutations (which have the complementary inversion set).

It turns out that there are double-exponentially many distinct lattice congruences of the weak order on S_n, and they generalize many known lattices, such as the Boolean lattice, the Tamari lattice [Tam62] (shown on the right hand side of Figure 10), and certain Cambrian lattices [Rea06, CP17]. This area of study has beautiful ramifications into groups, posets, polytopes, geometry, and combinatorics, and has been developed considerably in recent years, in particular thanks to Nathan Reading's works, summarized in [Rea12a, Rea16a, Rea16b].

5.2. **Jumping through lattice congruences.** For any lattice congruence $\equiv$ of the weak order on S_n, a *set of representatives* for the equivalence classes $S_n/\equiv$ is a subset $R_n \subseteq S_n$ such that for every equivalence class $X \in S_n/\equiv$, exactly one permutation is contained in R_n, i.e., $|X \cap R_n| = 1$. We let $X(\pi)$, $\pi \in S_n$, denote the equivalence class from $S_n/\equiv$ containing π. A meaningful definition of 'generating the lattice congruence' is to generate a set of representatives for its equivalence classes. We also require that any two successive representatives form a cover relation in the lattice quotient $S_n/\equiv$. This is what we achieve with the help of Algorithm J.

Theorem 22. *For every lattice congruence $\equiv$ of the weak order on S_n, there is a set of representatives $R_n \subseteq S_n$, such that Algorithm J generates a sequence $J(R_n) = \pi_1, \pi_2, \ldots$ of all permutations from R_n for which the equivalence classes $X(\pi_1), X(\pi_2), \ldots$ form a Hamilton path in the cover graph of the lattice quotient $S_n/\equiv$.*

For every lattice congruence $\equiv$, Pilaud and Santos [PS19] defined a polytope, called the *quotientope* for $\equiv$, whose skeleton is exactly the cover graph of the lattice quotient $S_n/\equiv$. These polytopes generalize many known polytopes, such as hypercubes, associahedra, permutahedra etc. The following result is an immediate corollary of Theorem 22, and it is illustrated in Figure 11.

Corollary 23. *For every lattice congruence $\equiv$ of the weak order on S_n, Algorithm J generates a Hamilton path on the skeleton of the corresponding quotientope.*

6. Acknowledgments

We thank Michael Albert, Mathilde Bouvel, Sergi Elizalde, Vít Jelínek, Sergey Kitaev, Vincent Vajnovszki, and Vincent Vatter for very insightful feedback on this work, for pointing out relevant references, and for sharing their knowledge about pattern-avoiding permutations. We also thank Jean Cardinal and Vincent Pilaud for several stimulating discussions about lattice congruences of the weak order on the symmetric group. Figure 11 in this paper was obtained by modifying and augmenting Figure 9 from [PS19], and the original source code for this figure was provided to us by Vincent Pilaud.

References

[AAB+13] M. H. Albert, M. D. Atkinson, M. Bouvel, N. Ruškuc, and V. Vatter. Geometric grid classes of permutations. *Trans. Amer. Math. Soc.*, 365(11):5859–5881, 2013.

[AB16] M. Albert and R. Brignall. 2 × 2 monotone grid classes are finitely based. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 1, 10, 2016.

[ABBM+13] A. Asinowski, G. Barequet, M. Bousquet-Mélou, T. Mansour, and R. Y. Pinter. Orders induced by segments in floorplans and (2-14-3, 3-41-2)-avoiding permutations. *Electron. J. Combin.*, 20(2):Paper 35, 43, 2013.

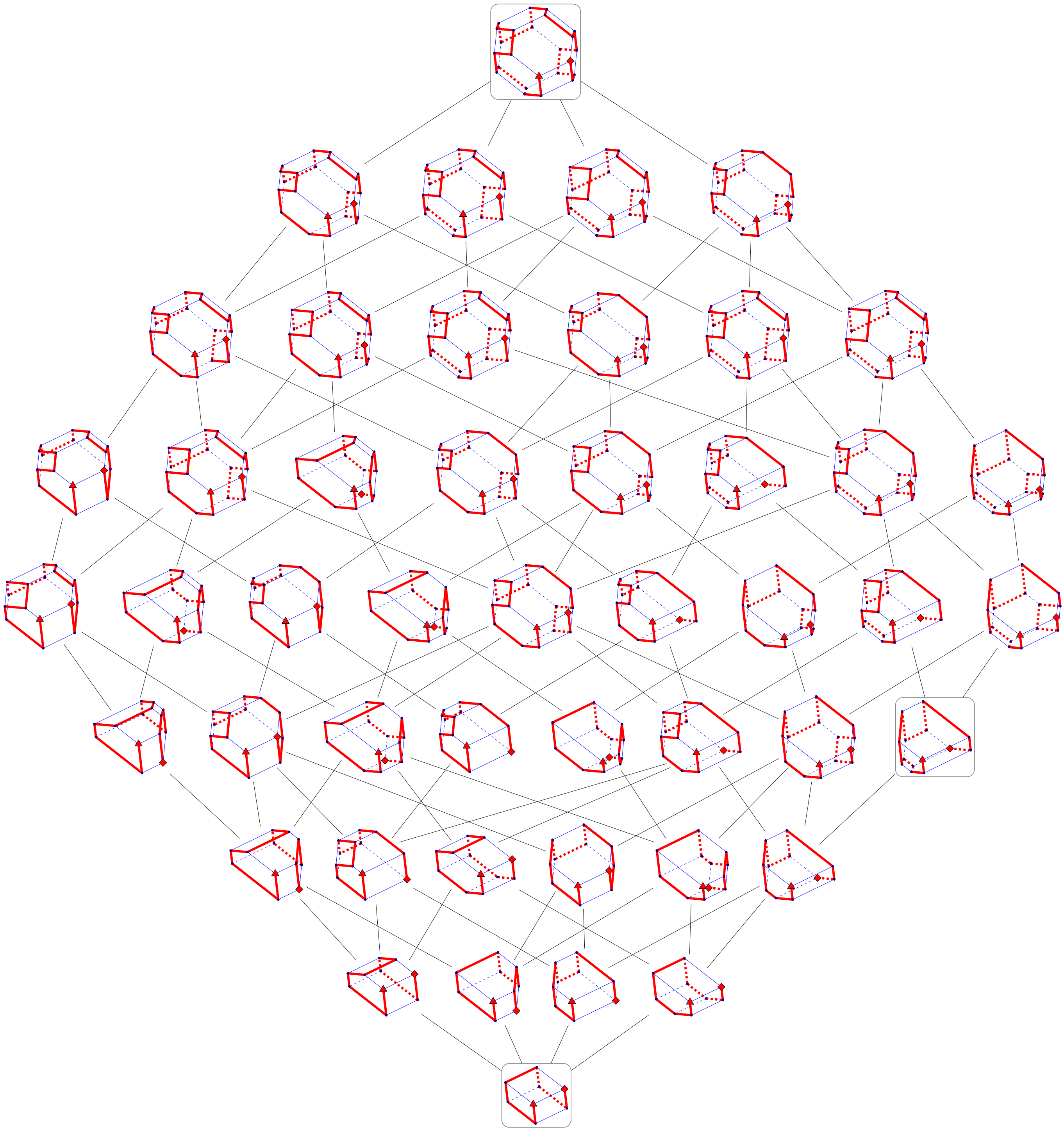


FIGURE 11. Lattice congruences of the weak order on S_4, ordered by refinement and realized as polytopes, where only the full-dimensional polytopes are shown. The figure shows the Hamilton path on each quotientope generated by Algorithm J, with the start and end vertex indicated by a triangle and diamond, respectively. Permutahedron, associahedron (one of four isomorphic variants) and hypercube are highlighted.

[ABP06] E. Ackerman, G. Barequet, and R. Y. Pinter. A bijection between permutations and floorplans, and its applications. *Discrete Appl. Math.*, 154(12):1674–1684, 2006.

[AF96] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996. First International Colloquium on Graphs and Optimization (GOI), 1992 (Grimentz).

[AKPV16] S. Avgustinovich, S. Kitaev, V. N. Potapov, and V. Vajnovszki. Gray coding cubic planar maps. *Theoret. Comput. Sci.*, 616:59–69, 2016.

[AN81] D. Avis and M. Newborn. On pop-stacks in series. *Utilitas Math.*, 19:129–140, 1981.

[Atk98] M. D. Atkinson. Permutations which are the union of an increasing and a decreasing subsequence. *Electron. J. Combin.*, 5:Research paper 6, 13, 1998.

[Bar08] J.-L. Baril. Efficient generating algorithm for permutations with a fixed number of excedances. *Pure Math. Appl. (PU.M.A.)*, 19(2-3):61–69, 2008.

[Bar09] J.-L. Baril. More restrictive Gray codes for some classes of pattern avoiding permutations. *Inform. Process. Lett.*, 109(14):799–804, 2009.

[Bar11] J.-L. Baril. Classical sequences revisited with permutations avoiding dotted pattern. *Electron. J. Combin.*, 18(1):Paper 178, 18, 2011.

[BBGP04] S. Bacchelli, E. Barcucci, E. Grazzini, and E. Pergola. Exhaustive generation of combinatorial objects by ECO. *Acta Inform.*, 40(8):585–602, 2004.

[BBL98] P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65(5):277–283, 1998.

[BC11] P. Brändén and A. Claesson. Mesh patterns and the expansion of permutation statistics as sums of permutation patterns. *Electron. J. Combin.*, 18(2):Paper 5, 14, 2011.

[BDLPP99] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects. *J. Differ. Equations Appl.*, 5(4-5):435–490, 1999.

[BER76] J. R. Bitner, G. Ehrlich, and E. M. Reingold. Efficient generation of the binary reflected Gray code and its applications. *Comm. ACM*, 19(9):517–521, 1976.

[BEV18] R. Brignall, M. Engen, and V. Vatter. A counterexample regarding labelled well-quasi-ordering. *Graphs Combin.*, 34(6):1395–1409, 2018.

[BGPP07] A. Bernini, E. Grazzini, E. Pergola, and R. Pinzani. A general exhaustive generation algorithm for Gray structures. *Acta Inform.*, 44(5):361–376, 2007.

[Bil13] S. Billey. Permutation patterns for k-vexillary permutations, 2013. `https://sites.math.washington.edu/~billey/papers/k.vex.html`.

[BL10] A. Burstein and I. Lankham. Restricted patience sorting and barred pattern avoidance. In *Permutation patterns*, volume 376 of *London Math. Soc. Lecture Note Ser.*, pages 233–257. Cambridge Univ. Press, Cambridge, 2010.

[BMCDK10] M. Bousquet-Mélou, A. Claesson, M. Dukes, and S. Kitaev. $(2 + 2)$-free posets, ascent sequences and pattern avoiding permutations. *J. Combin. Theory Ser. A*, 117(7):884–909, 2010.

[Bón97] M. Bóna. Exact enumeration of 1342-avoiding permutations: a close link with labeled trees and planar maps. *J. Combin. Theory Ser. A*, 80(2):257–272, 1997.

[Bón12] M. Bóna. *Combinatorics of permutations*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, second edition, 2012. With a foreword by Richard Stanley.

[BP14] S. Billey and B. Pawlowski. Permutation patterns, Stanley symmetric functions, and generalized Specht modules. *J. Combin. Theory Ser. A*, 127:85–120, 2014.

[BS00] E. Babson and E. Steingrímsson. Generalized permutation patterns and a classification of the Mahonian statistics. *Sém. Lothar. Combin.*, 44:Art. B44b, 18, 2000.

[CF18] J. Cardinal and S. Felsner. Topological drawings of complete bipartite graphs. *J. Comput. Geom.*, 9(1):213–246, 2018.

[CK08] A. Claesson and S. Kitaev. Classification of bijections between 321- and 132-avoiding permutations. In *20th Annual International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2008)*, Discrete Math. Theor. Comput. Sci. Proc., AJ, pages 495–506. Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2008.

[CKS09] A. Claesson, S. Kitaev, and E. Steingrímsson. Decompositions and statistics for $\beta(1, 0)$-trees and nonseparable permutations. *Adv. in Appl. Math.*, 42(3):313–328, 2009.

[CP17] G. Chatel and V. Pilaud. Cambrian Hopf algebras. *Adv. Math.*, 311:598–633, 2017.

[CSS18] J. Cardinal, V. Sacristán, and R. I. Silveira. A note on flips in diagonal rectangulations. *Discrete Math. Theor. Comput. Sci.*, 20(2):Paper No. 14, 22, 2018.

[DFMV08] W. M. B. Dukes, M. F. Flanagan, T. Mansour, and V. Vajnovszki. Combinatorial Gray codes for classes of pattern avoiding permutations. *Theoret. Comput. Sci.*, 396(1-3):35–49, 2008.

[DG12] P. Diaconis and R. Graham. *Magical mathematics*. Princeton University Press, Princeton, NJ, 2012. The mathematical ideas that animate great magic tricks, With a foreword by Martin Gardner.

[DGG98] S. Dulucq, S. Gire, and O. Guibert. A combinatorial proof of J. West's conjecture. *Discrete Math.*, 187(1-3):71–96, 1998.

[DGW96] S. Dulucq, S. Gire, and J. West. Permutations with forbidden subsequences and nonseparable planar maps. In *Proceedings of the 5th Conference on Formal Power Series and Algebraic Combinatorics (Florence, 1993)*, volume 153, pages 85–103, 1996.

[DMR10] E. Deutsch, E. Munarini, and S. Rinaldi. Skew Dyck paths. *J. Statist. Plann. Inference*, 140(8):2191–2203, 2010.

[DRT14] P. T. Do, D. Rossin, and T. T. H. Tran. Permutations weakly avoiding barred patterns and combinatorial bijections to generalized Dyck and Motzkin paths. *Discrete Math.*, 320:40–50, 2014.

[DTV19] P. T. Do, T. T. H. Tran, and V. Vajnovszki. Exhaustive generation for permutations avoiding (colored) regular sets of patterns. *Discrete Applied Mathematics*, 2019.

[Eld06] M. Elder. Permutations generated by a stack of depth 2 and an infinite stack in series. *Electron. J. Combin.*, 13(1):Research Paper 68, 12, 2006.

[Eli07] S. Elizalde. Generating trees for permutations avoiding generalized patterns. *Ann. Comb.*, 11(3-4):435–458, 2007.

[Eli11] S. Elizalde. The X-class and almost-increasing permutations. *Ann. Comb.*, 15(1):51–68, 2011.

[FMSD19] A. Fink, K. Mészáros, and A. St. Dizier. Zero-one Schubert polynomials. `https://arxiv.org/abs/1903.10332`, 2019.

[FS09] P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[GM14] S. Guillemot and D. Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–101. ACM, New York, 2014.

[GP16] S. Garrabrant and I. Pak. Permutation patterns are hard to count. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 923–936. ACM, New York, 2016.

[Gra53] F. Gray. Pulse code communication, 1953. March 17, 1953 (filed Nov. 1947). U.S. Patent 2,632,058.

[GW96] I. P. Goulden and J. West. Raney paths and a combinatorial relationship between rooted nonseparable planar maps and two-stack-sortable permutations. *J. Combin. Theory Ser. A*, 75(2):220–242, 1996.

[HV06] S. Huczynska and V. Vatter. Grid classes and the Fibonacci dichotomy for restricted permutations. *Electron. J. Combin.*, 13(1):Research Paper 54, 14, 2006.

[Jer03] M. Jerrum. *Counting, sampling and integrating: algorithms and complexity*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, 2003.

[JK17] V. Jelínek and J. Kynčl. Hardness of permutation pattern matching. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–396. SIAM, Philadelphia, PA, 2017.

[Joh63] S. Johnson. Generation of permutations by adjacent transposition. *Math. Comp.*, 17:282–285, 1963.

[Kay76] R. Kaye. A Gray code for set partitions. *Information Processing Lett.*, 5(6):171–173, 1976.

[Kit05] S. Kitaev. Partially ordered generalized patterns. *Discrete Math.*, 298(1-3):212–229, 2005.

[Kit11] S. Kitaev. *Patterns in permutations and words*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, 2011. With a foreword by Jeffrey B. Remmel.

[Knu98] D. E. Knuth. *The art of computer programming. Vol. 3*. Addison-Wesley, Reading, MA, 1998. Sorting and searching, Second edition [of MR0445948].

[Knu11] D. E. Knuth. *The art of computer programming. Vol. 4A. Combinatorial algorithms. Part 1*. Addison-Wesley, Upper Saddle River, NJ, 2011.

[Koz19] L. Kozma. Faster and simpler algorithms for finding large patterns in permutations. `https://arxiv.org/abs/1902.08809`, 2019.

[Lan07] I. P. Lankham. *Patience sorting and its generalizations*. ProQuest LLC, Ann Arbor, MI, 2007. Thesis (Ph.D.)–University of California, Davis.

[LR12] S. Law and N. Reading. The Hopf algebra of diagonal rectangulations. *J. Combin. Theory Ser. A*, 119(3):788–824, 2012.

[LS85] A. Lascoux and M.-P. Schützenberger. Schubert polynomials and the Littlewood-Richardson rule. *Lett. Math. Phys.*, 10(2-3):111–124, 1985.

[LS09] Y. Li and J. Sawada. Gray codes for reflectable languages. *Inform. Process. Lett.*, 109(5):296–300, 2009.

[LvBR93] J. M. Lucas, D. R. van Baronaigien, and F. Ruskey. On rotations and the generation of binary trees. *J. Algorithms*, 15(3):343–366, 1993.

[NZ96] J. Noonan and D. Zeilberger. The enumeration of permutations with a prescribed number of "forbidden" patterns. *Adv. in Appl. Math.*, 17(4):381–407, 1996.

[oei19] OEIS Foundation Inc. The on-line encyclopedia of integer sequences, 2019. `http://oeis.org`.

[Par09] R. Parviainen. Wilf classification of bi-vincular permutation patterns. `https://arxiv.org/abs/0910.5103`, 2009.

[PS19] V. Pilaud and F. Santos. Quotientopes. *Bull. Lond. Math. Soc.*, 51:406–420, 2019.

[Pud08] L. K. Pudwell. *Enumeration schemes for pattern-avoiding words and permutations*. ProQuest LLC, Ann Arbor, MI, 2008. Thesis (Ph.D.)–Rutgers The State University of New Jersey - New Brunswick.

[Pud10] L. K. Pudwell. Enumeration schemes for permutations avoiding barred patterns. *Electron. J. Combin.*, 17(1):Research Paper 29, 27, 2010.

[Rea06] N. Reading. Cambrian lattices. *Adv. Math.*, 205(2):313–353, 2006.

[Rea12a] N. Reading. From the Tamari lattice to Cambrian lattices and beyond. In *Associahedra, Tamari lattices and related structures*, volume 299 of *Prog. Math. Phys.*, pages 293–322. Birkhäuser/Springer, Basel, 2012.

[Rea12b] N. Reading. Generic rectangulations. *European J. Combin.*, 33(4):610–623, 2012.

[Rea16a] N. Reading. Finite Coxeter groups and the weak order. In *Lattice theory: special topics and applications. Vol. 2*, pages 489–561. Birkhäuser/Springer, Cham, 2016.

[Rea16b] N. Reading. Lattice theory of the poset of regions. In *Lattice theory: special topics and applications. Vol. 2*, pages 399–487. Birkhäuser/Springer, Cham, 2016.

[RSW12] F. Ruskey, J. Sawada, and A. Williams. Binary bubble languages and cool-lex order. *J. Combin. Theory Ser. A*, 119(1):155–169, 2012.

[Sav97] C. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997.

[Sta94] Z. E. Stankova. Forbidden subsequences. *Discrete Math.*, 132(1-3):291–316, 1994.

[SV14] R. Smith and V. Vatter. A stack and a pop stack in series. *Australas. J. Combin.*, 58:157–171, 2014.

[SW12] J. Sawada and A. Williams. Efficient oracles for generating binary bubble languages. *Electron. J. Combin.*, 19(1):Paper 42, 20, 2012.

[Tam62] D. Tamari. The algebra of bracketings and their enumeration. *Nieuw Arch. Wisk. (3)*, 10:131–146, 1962.

[Ten18] B. Tenner. Database of permutation pattern avoidance, 2018. `https://math.depaul.edu/bridget/patterns.html`.

[Tro62] H. F. Trotter. Algorithm 115: Perm. *Commun. ACM*, 5(8):434–435, 1962.

[Úlf11] H. Úlfarsson. A unification of permutation patterns related to Schubert varieties. *Pure Math. Appl. (PU.M.A.)*, 22(2):273–296, 2011.

[Vaj10] V. Vajnovszki. Generating involutions, derangements, and relatives by ECO. *Discrete Math. Theor. Comput. Sci.*, 12(1):109–122, 2010.

[Vel03] A. Vella. Pattern avoidance in permutations: linear and cyclic orders. *Electron. J. Combin.*, 9(2):Research paper 18, 43, 2002/03. Permutation patterns (Otago, 2003).

[VV11] V. Vajnovszki and R. Vernay. Restricted compositions and permutations: from old to new Gray codes. *Inform. Process. Lett.*, 111(13):650–655, 2011.

[Wat07] S. D. Waton. *On permutation classes defined by token passing networks, gridding matrices and pictures: three flavours of involvement.* PhD thesis, University of St Andrews, 2007.

[Wes90] J. West. *Permutations with forbidden subsequences and stack-sortable permutations*. ProQuest LLC, Ann Arbor, MI, 1990. Thesis (Ph.D.)–Massachusetts Institute of Technology.

[Wil13] A. Williams. The greedy Gray code algorithm. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, pages 525–536, 2013.

[WY06] A. Woo and A. Yong. When is a Schubert variety Gorenstein? *Adv. Math.*, 207(1):205–220, 2006.

[XCU10] L. Xiang, K. Cheng, and K. Ushijima. Efficient generation of Gray codes for reflectable languages. In *Computational Science and Its Applications - ICCSA 2010, International Conference, Fukuoka, Japan, March 23-26, 2010, Proceedings, Part IV*, pages 418–426, 2010.

[YCCG03] B. Yao, H. Chen, C.-K. Cheng, and R. L. Graham. Floorplan representations: Complexity and connections. *ACM Trans. Design Autom. Electr. Syst.*, 8(1):55–80, 2003.

[Zei92] D. Zeilberger. A proof of Julian West's conjecture that the number of two-stack-sortable permutations of length n is $2(3n)!/((n+1)!(2n+1)!)$. *Discrete Math.*, 102(1):85–93, 1992.