FPSAC 2012, Nagoya, Japan DMTCS proc. **AR**, 2012, 595–606

Sorting and preimages of pattern classes

Anders Claesson[1] and Henning Úlfarsson[2]

[1]*Department of Computer and Information Sciences, University of Strathclyde, Glasgow G1 1XH, UK*
[2]*School of Computer Science, Reykjavik University, Menntavegi 1, 101 Reykjavík, Iceland*

Abstract. We introduce an algorithm to determine when a sorting operation, such as stack-sort or bubble-sort, outputs a given pattern. The algorithm provides a new proof of the description of West-2-stack-sortable permutations, that is permutations that are completely sorted when passed twice through a stack, in terms of patterns. We also solve the long-standing problem of describing West-3-stack-sortable permutations. This requires a new type of generalized permutation pattern we call a decorated pattern.

Résumé. On introduit un algorithme qui détermine quand un opérateur de tri (tels que le tri par une pile, ou le tri-bulle) produit un motif donné en sortie. Cet algorithme fournit une nouvelle preuve de la caractérisation des permutations 2-triables au sens de West (c'est-à-dire des permutations qui sont triées complètement par deux passages dans une pile) par des motifs interdits. On résout aussi le problème longtemps ouvert de la caractérisation des permutations 3-triables au sens de West. Ceci demande de définir un nouveau type de motifs généralisés, appelé motif décoré.

Keywords: Permutation Patterns, Sorting algorithms

1 Introduction

The set of permutations of $\{1,\ldots,n\}$ is denoted $\mathcal{S}_n$. Permutations will be written in one-line notation, and the identity permutation $12\cdots n$ will be denoted id_n, or just id if n is understood from context.

In the 1970's Knuth [Knu75] initiated the study of sorting and pattern avoidance in permutations. He considered the problem of sorting a permutation by passing it through a stack. A *stack* is a last in, first out data structure with two fundamental operations: the *push* operation moves an item from the input to the top of the stack; the *pop* operation moves an item from the top of the stack to the output. Consider trying to sort the permutation 231 by a stack, as shown in Figure 1.

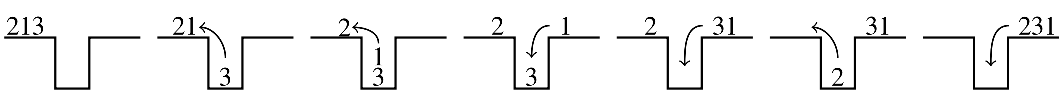


Fig. 1: Trying, and failing, to sort 231 with a stack. The figure is read from right to left

Note that we always want the elements in the stack to be increasing, from the top, since otherwise it would be impossible for the output to be sorted. We failed to sort the permutation in one pass and therefore say that it is not *stack-sortable*, which Knuth showed is equivalent to containg 231 as a pattern. We will reprove this in Theorem 3.1. Several variations on Knuth's original problem have been considered, see Bóna [Bón03] for a survey. In this paper we consider repeatedly passing a permutation through a stack, while keeping the elements on the stack in increasing order from top to bottom. We also consider the bubble-sort operator. We introduce a new method for finding patterns in a permutation that will cause these sorting devices to output a given pattern, and use this to solve the long-standing problem of describing West-3-stack-sortable permutations. If the given pattern is classical (defined below) we show that the mesh patterns introduced by Brändén and the first author [BC11] suffice, but if the given pattern is itself a mesh pattern we will need to introduce a new kind of generalized pattern we call a *decorated* pattern.

In Section 4 we describe an algorithm that given a classical pattern p produces a finite list of (marked) mesh patterns P such that $S^{-1}(\mathrm{Av}(p)) = \mathrm{Av}(P)$, where S is the stack-sort operator. This algorithm automates proving some of the statements in the previous sections and can be extended to cover the bubble-sort operator as well.

We hope that our work is a step towards the general comprehension of the stack-sort operator and other similar operators. This extended abstract is based on two papers of the authors [Ú11a, CU11].

2 Generalized permutation patterns

A *standardization* of a list of numbers is another list of the same length such that the smallest letter in the original list has been replaced with 1, the second smallest with 2, *etc*. The standardization of 5371 is 3241. A *classical* (*permutation*) *pattern* is a permutation p in $\mathcal{S}_k$. A permutation π in $\mathcal{S}_n$ *contains*, or *has an occurrence of* the pattern p if there are indices $1 \leq i_1 < \cdots < i_k \leq n$ such that the standardization of $\pi(i_1)\cdots\pi(i_k)$ equals the pattern p. If a permutation does not contain a pattern p we say it *avoids* p. The permutation $\pi = 526413$ contains the pattern $p = 132$, and has three occurrences of it, in the subsequences 264, 263 and 243. We can draw the *graph* of the permutation by graphing the coordinates $(i, \pi(i))$ on a grid. For example the permutation π above is shown in Figure 2 where we have circled the occurrences of the pattern p.

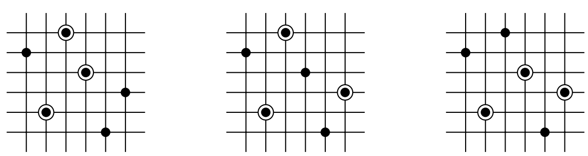
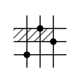


Fig. 2: The permutation 526413 and three occurrences of the pattern 132

The same permutation avoids 123, since we cannot find an increasing subsequence of length three.

2.1 Mesh patterns and barred patterns

Mesh patterns were introduced in [BC11]. We review them via an example. The mesh pattern  occurs in a permutation if we can find the underlying classical pattern 132 positioned in such a way that the shaded regions are not occupied by other entries in the permutation. Another way of writing a mesh pattern is to give the underlying classical pattern, followed by the set of shaded boxes, labelled by their lower left corner (the left-most box in the bottom-most row being $(0, 0)$). This mesh pattern is $(132, \{(0, 2), (1, 2), (2, 2)\})$.

Consider the permutation 526413. From above we know that the classical pattern has three occurrences in this permutation. In Figure 3 one can see that just one of these satisfies the additional requirement that there be no additional entries in the shaded region "between and to the left of the 3 and the 2".

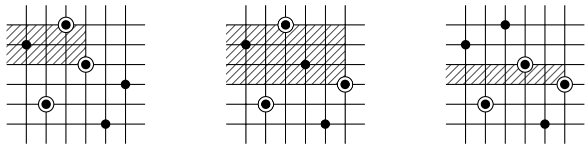
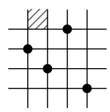
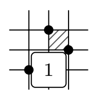


Fig. 3: The permutation 526413 and one occurrences of the mesh pattern $(132, \{(0,2),(1,2),(2,2)\})$

Barred patterns, introduced by West [Wes90], are classical patterns with bars over some of the entries. Such a pattern is contained in a permutation if the standardization of the unbarred entries is contained in the permutation in such a way that they are not part of an occurrence of the whole pattern. The permutation 5264173 contains the barred pattern $3\bar{5}241$, in the subsequence 5473, since that is an occurrence of 3241 that is not part of an occurrence of 35241. In [BC11] it was shown that any barred pattern with one barred entry is a mesh pattern. The barred pattern we discussed here is in fact the mesh pattern below.



2.2 Marked mesh patterns and decorated patterns

Marked mesh patterns were introduced by the second author in [Ú11b] and give finer control over whether a certain region in a permutation is allowed to contain elements, and if so, how many. Again we just give an example. Consider the marked mesh pattern  . The meaning of the 1 in the region containing boxes $(1,0)$, $(1,1)$, $(2,0)$ and $(2,1)$ is that this region must contain *at least* one entry. In Figure 4 we see that there is exactly one occurrence of this mesh pattern in the permutation 526413.

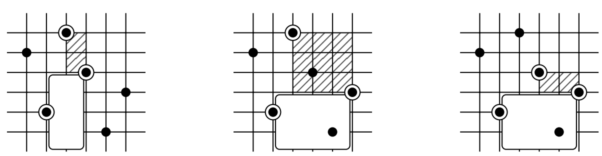
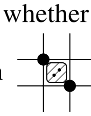


Fig. 4: The permutation 526413 and one occurrence of a marked mesh pattern

Marked mesh patterns will be useful when we need to add elements into an existing pattern to ensure other elements are popped by a particular sorting device. Below we will need even finer control over what is allowed inside a particular region in a pattern. We will need to control whether the entries in the region contain a particular pattern. Consider for example the decorated pattern  . The decorated region in the middle signifies that an occurrence of this pattern should be an occurrence of the underlying classical pattern 21 that additionally does not have entries in the region that contain the pattern 12 – or equivalently – whatever is in that region must be in descending order, from left to right. In Figure 5 there is an occurrence of the decorated pattern on the left and on the right we have an occurrence of the classical pattern 21 that does not satisfy the requirements of the decorated region. Below we state the formal definition of a decorated pattern.

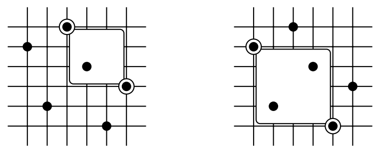


Fig. 5: The permutation 526413 and one occurrences of a decorated pattern

For integers $a < b$ we use $[\![a, b]\!]$ to denote the set $\{a, a+1, \dots, b\}$.

Definition 2.1 *A* decorated pattern $(p, \mathcal{C})$ *of length k consists of a classical pattern p of length k and a collection $\mathcal{C}$ which contains pairs (C, q) where C is a subset of the square $[\![0, k]\!] \times [\![0, k]\!]$ and q is some pattern, possibly another decorated pattern. An occurrence of $(p, \mathcal{C})$ in a permutation π is a subset ω of the diagram $G(\pi) = \{(i, \pi(i)) \mid 1 \leq i \leq n\}$ such that there are order-preserving injections $\alpha, \beta : [\![1, k]\!] \to [\![1, n]\!]$ satisfying two conditions:*

1. *$\omega = \{(\alpha(i), \beta(j)) : (i, j) \in G(p)\}$.*
2. *Let $R_{ij} = [\![\alpha(i) + 1, \alpha(i+1) - 1]\!] \times [\![\beta(j) + 1, \beta(j+1) - 1]\!]$, with $\alpha(0) = 0 = \beta(0)$ and $\alpha(k+1) = n+1 = \beta(k+1)$. For each pair (C, q) we let $C' = \bigcup_{(i,j) \in C} R_{ij}$ and require that $C' \cap G(\pi)$ avoids q.*

3 Finding preimages of patterns

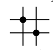
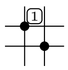
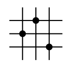
In this section we define a method for describing patterns that are guaranteed to produce a given pattern in a permutation after it is sorted by a stack or with the bubble-sort operator.

3.1 The stack-sort operator

For a permutation π we will denote with $S(\pi)$ the image of π after it is passed once through a stack. Note that a permutation π in $\mathcal{S}_n$ is stack-sortable if and only if $S(\pi) \in \mathrm{Av}_n(21)$, where $\mathrm{Av}_n(21)$ is the set of permutations of length n that avoid 21. Of course $\mathrm{Av}_n(21) = \{\mathrm{id}\}$ but framing the definition like this leads to a generalization: Given a pattern p, what conditions need to be put on π such that $S(\pi) \in \mathrm{Av}_n(p)$. Exploring a sorting operator from this angle was first done by Albert *et al.* [AAB+11]; they, however, studied the bubble-sort operator rather than the stack-sort operator.

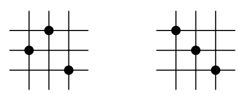
Below we will call permutations such that $S^k(\pi) = \mathrm{id}$, *West-k-stack-sortable* permutations, since West considered this generalization from the case of one stack first. Note that for $k > 1$ these permutations are different from the *k-stack-sortable* permutations, which are the permutations that can be sorted by using k stacks in series without the requirement that the elements on the stack are increasing from top to bottom. For example, the permutation 2341 is not West-2-stack-sortable, but if we put the entries 2, 3, 4 onto the first stack, pass 1 all the way to the end, and then use the second stack to sort $2, 3, 4$ we end up with 1234. So 2341 is 2-stack-sortable.

The basic idea behind the method we are about to describe is that $S(\pi)$ has an occurrence of a pattern p of length k if and only if the k elements in this occurrence were present in π as some kind of pattern before we sorted. We start by showing how this idea allows us to describe stack-sortable permutations as well as West-2-stack-sortable permutations.

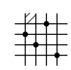
We know that π is not sorted by the stack if and only if $S(\pi)$ contains the classical pattern 21. Therefore consider a particular occurrence of this pattern in $S(\pi)$. Before sorting, the elements in this occurrence must have formed the pattern $21 =$  in π. In order to remain in this order the element corresponding to 2 must be popped off the stack by a larger element before the element corresponding to 1 enters. Thus the box $(1, 2)$ must have at least one element and we have an occurrence of the marked mesh pattern  which is equivalent to the pattern . We have therefore reproven Knuth's result.

Theorem 3.1 ([Knu75]) *A permutation is stack-sortable if and only if it avoids* 231.

We can similarly reprove West's result on West-2-stack-sortable permutations, *i.e.*, permutations π such that $S^2(\pi) = \mathrm{id}$. By Knuth's result we know that π will be sorted by two passes through the stack if and only if $S(\pi)$ avoids the pattern 231. An occurrence of 231 must have formed either of the patterns



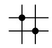
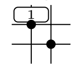
in π. For the elements in the pattern on the left to stay in the same order as they pass through the stack we must have an element in the box $(2, 3)$ to pop the element corresponding to 3 out of the stack before the smallest element enters. Now, the opposite happens for the pattern on the right. The 3 must stay on the stack until 2 enters, so there can be no elements in the box $(1, 3)$. Then both 2 and 3 must leave the stack before 1 enters. Thus the patterns above become the two marked mesh patterns on the left below. These are more naturally written as the two mesh patterns on the right.

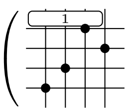
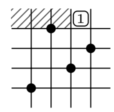
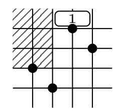
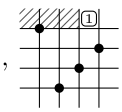
By a lemma of Hilmarsson, *et al.* [HJS$^+$11], the last pattern on the right is equivalent to , in the sense that a permutation either contains both patterns or avoids both. (For these two patterns it is also easy to see directly that they are equivalent.) As mentioned above this pattern is another representation of the barred pattern $3\bar{5}241$. Thus we have re-derived West's result.

Theorem 3.2 ([Wes90]) *A permutation is West-*2*-stack-sortable if and only if it avoids* 2341 *and* $3\bar{5}241$.

3.2 The bubble-sort operator

The bubble-sort operator swaps adjacent entries in a permutation if the left entry is larger than the right entry. Let $B(\pi)$ denote the output of one pass of bubble-sort on π. For instance, $B(521634) = 215346$. A modification of the method above works equally well for B. We see that $B(\pi)$ contains the pattern 21

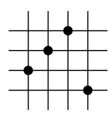
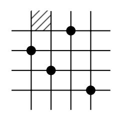
if and only if π contains $21 =$ . To make sure that these elements stay in this order, we either need a large element in front of 2, which would mean 2 would never be moved; or a large element in between 2 and 1 that will stop 2 from moving past 1. We get the marked mesh pattern , which is equivalent to the two classical patterns 231 and 321. Thus, $B(\pi) = \mathrm{id}$ if and only if π avoids 231 and 321, as was first made explicit by Albert, *et al.* [AAB$^+$11]. In the same paper the authors show that for any classical pattern p with at least three left-to-right maxima, the third of which is not the final symbol of p, the set $B^{-1}(\mathrm{Av}(p))$ is not described by classical patterns. We consider the smallest example of such a pattern, $p = 1243$. For a proof of the following proposition see [Ú11a, Proposition 3.3].

Proposition 3.3 $B^{-1}(\mathrm{Av}(1243)) = \mathrm{Av}$ (, , , ).

Note that all the patterns can be expanded to mesh patterns, but at the cost of having more patterns.

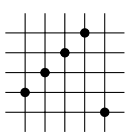
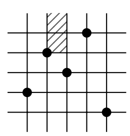
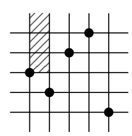
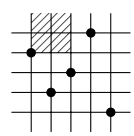
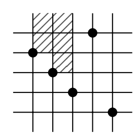
3.3 West-3-stack-sortable permutations

We now turn to West-3-stack-sortable permutations, *i.e.*, permutations π such that $S^3(\pi) = \mathrm{id}$. By West's result (Theorem 3.2) $S^3(\pi) = \mathrm{id}$ if and only if $S(\pi)$ avoids these two patterns:

$W_1 =$  $W_2 =$  (3.1)

We will use the same method as we did above, but when we consider the pattern on the right, the shaded box will cause some complications and the decorated patterns introduced above will be necessary.

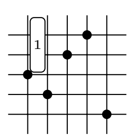
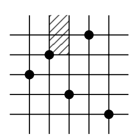
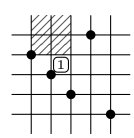
Lemma 3.4 *An occurrence of* W_1 *in* $S(\pi)$ *comes from exactly one of the patterns below in* π.

$I_1 =$  $I_2 =$  $I_3 =$  $I_4 =$  $I_5 =$ 

Proof: Use Algorithm 1 from Section 4. See also [Ú11a, Lemma 4.1]. □

We now consider the pattern W_2, but without the shading.

Lemma 3.5 *An occurrence of* 3241 *in* $S(\pi)$ *comes from exactly one of the patterns below in* π.

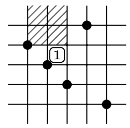
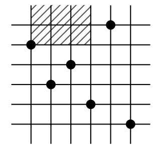
$j_1 =$  $j_2 =$  $j_3 =$ 

Proof: Use Algorithm 1 from Section 4. See also [Ú11a, Lemma 4.2]. □

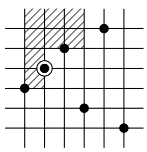
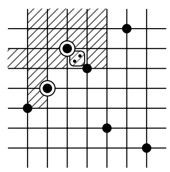
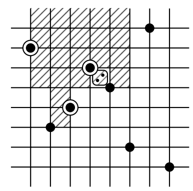
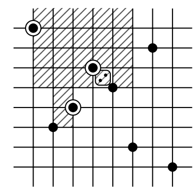
We now consider what additional conditions cause the patterns j_1, j_2 and j_3 in the lemma to give the correct shading in the pattern W_2. We express these conditions in the next lemma and two propositions.

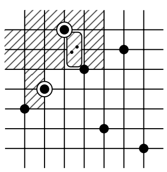
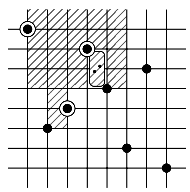
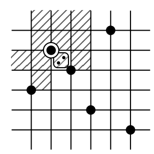
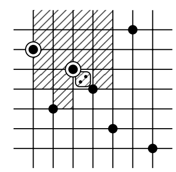
Lemma 3.6 *An occurrence of j_3 in a permutation π will become an occurrence of W_2 in $S(\pi)$.*

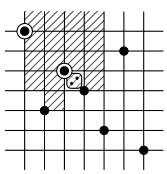
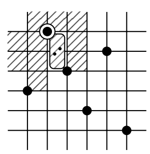
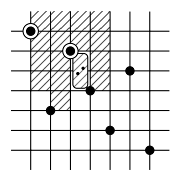
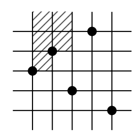
We leave the proof to the reader. We rename the pattern J_3 and note that it can also be expanded into a mesh pattern.

$J_3 =$  $=$ 

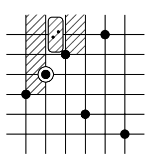
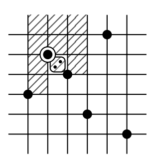
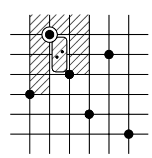
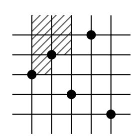
Proposition 3.7 *An occurrence of j_2 in a permutation π becomes an occurrence of W_2 in $S(\pi)$ if and only if it is part of one of the patterns below, where elements that have been added to j_2 are circled.*

$J_{2,1} =$  $J_{2,2} =$  $J_{2,3} =$  $J_{2,4} =$ 

$J_{2,5} =$  $J_{2,6} =$  $J_{2,7} =$  $J_{2,8} =$ 

$J_{2,9} =$  $J_{2,10} =$  $J_{2,11} =$  $J_{2,12} =$ 

Proof: To ensure that there are no elements in the shaded box in W_2 we must look at the element that pops 3 in j_2. There are four different possibilities.

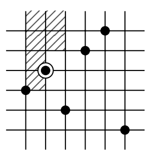
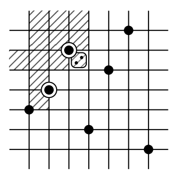
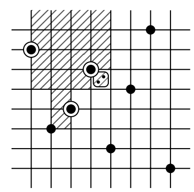
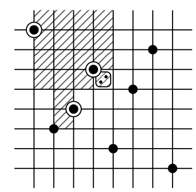
$j_{2,1} =$  $j_{2,2} =$  $j_{2,3} =$  $j_{2,4} =$ 

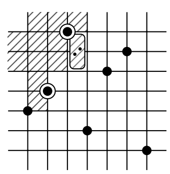
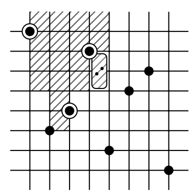
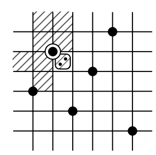
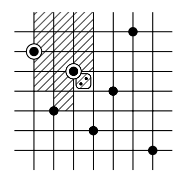
We explain the shadings and the decoration of the pattern $j_{2,2}$ as the others are similar. For this pattern, the size of the element that popped 3 from the stack was in-between 4 and 5. Since this was the element that popped 3 there can be no elements in boxes $(1,3), \ldots, (1,6)$. The boxes $(2,5)$ and $(2,6)$ cannot contain an element, since that would pop the element we just added (the 5 in $j_{2,2}$) and this element would land in the shaded box in W_2. Now consider the decorated box $(2,4)$. It can contain elements, but none of them are allowed to leave the stack prior to 4 being pushed on, since any one of them would then land in the shaded box in W_2. Elements in this region must then be in descending order.

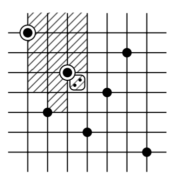
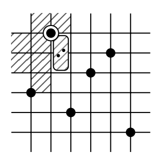
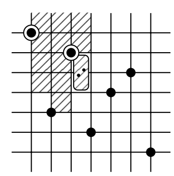
We must make sure that elements that arrived on the stack prior to 3 are not popped into the shaded box in W_2. We only consider the pattern $j_{2,2}$ here. If there are elements from box $(0, 4)$ still on the stack when 3 is put on they will be popped by 5 and will land in the shaded box in W_2. We must therefore have this box empty, or an element in box $(0, 5)$ or $(0, 6)$ that pops everything before 3 is pushed on. We get the three patterns $J_{2,7}, J_{2,8}, J_{2,9}$. □

We now consider what constraints must be imposed on the pattern j_1 to get W_2 after sorting.

Proposition 3.8 *An occurrence of j_1 in a permutation π becomes an occurrence of W_2 in $S(\pi)$ if and only if it is part of one of the patterns below, where elements that have been added to j_1 are circled.*

$J_{1,1} =$ 
$J_{1,2} =$ 
$J_{1,3} =$ 
$J_{1,4} =$ 

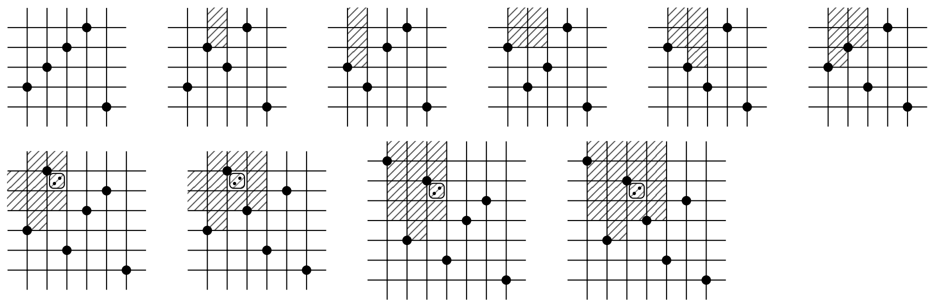
$J_{1,5} =$ 
$J_{1,6} =$ 
$J_{1,7} =$ 
$J_{1,8} =$ 

$J_{1,9} =$ 
$J_{1,10} =$ 
$J_{1,11} =$ 

Proof: The proof of this proposition is similar to the proof of Proposition 3.7 and therefore omitted. □

Taken together, Lemmas 3.4 and 3.6, with Propositions 3.7 and 3.8 produce a list of 29 patterns describing permutations that are not West-3-stack-sortable. We can simplify this list considerably by observing that the patterns $J_{1,1}, \ldots, J_{1,6}$ all imply containment of I_1 so we can remove them. Further simplifications of this sort can be made. Also, by considering what happens with the decorated patterns when they contain a certain number of elements in the decorated region the list can be simplified even further. See [Ú11a, Theorem 4.6] for the details.

Theorem 3.9 *A permutation is West-3-stack-sortable if and only if it avoids the decorated patterns*



Note that each of the decorated patterns in the theorem is equivalent to an infinite family of mesh patterns.

4 An algorithm

We shall now automate proving statements such as Theorem 3.2. More precisely we shall provide an algorithm that, given a classical pattern p, produces a finite list of (marked) mesh patterns P such that

$$S^{-1}(\mathrm{Av}(p)) = \mathrm{Av}(P).$$

The algorithm can be modified for the bubble-sort operator to prove statements such as Proposition 3.3. Given the classical pattern p we identify what orderings of the letters in p are possible prior to sorting, and produce a list, denoted $\mathsf{cand}(p)$, of candidates which themselves are classical patterns.

Proposition 4.1 *Let p be a permutation of a finite set of integers, and let the largest letter of p be $m = \max(p)$. Write $p = \alpha m \beta$ and $\alpha = a_1 a_2 \cdots a_i$. Then*

$$\mathsf{cand}(p) \;=\; \bigcup_{j=0}^{i} \{\, \gamma m \delta : \gamma \in \mathsf{cand}(a_1 a_2 \cdots a_j),\ \delta \in \mathsf{cand}(a_{j+1} \cdots a_i \beta) \,\}$$

contains all classical patterns that can become p after one pass of stack-sort.

This proposition gives a recursive algorithm for computing $\mathsf{cand}(p)$. Recall that an *inversion* in a permutation is an occurrence of the classical pattern 21, while a *non-inversion* is an occurrence of 12. Note that if two elements in p are part of an inversion, they must also be part of an inversion in all patterns in $\mathsf{cand}(p)$. Non-inversion in p place no restrictions on the patterns in $\mathsf{cand}(p)$.

Proof: The idea behind the proof is that letters can only be moved to the left and small letters are stopped by larger letters. This implies that after the largest letter, m, has been moved to a particular location we can recurse on what is remaining to the left and the right. $\square$

Note that $\mathsf{cand}(132) = \{321, 312, 132\}$. However, it is easy to check that there is no way that 321 can become 132 after sorting. Candidates like this one are removed in lines 9–11 in Algorithm 1 below.

Example 4.2 *For the pattern $p = 3241$, considered in Lemma 3.5, we have $m = 4$, $\alpha = 32$, $\beta = 1$ and $i = 2$. When $j = 0$, γ is the empty word and we get the set $\{4\delta \colon \delta \in \mathsf{cand}(321)\}$. It is easy to verify that*

$\mathsf{cand}(321) = \{321\}$*, so we get the pattern* 4321*. When* $j = 1$ *we get the set* $\{\gamma 4\delta \colon \gamma \in \mathsf{cand}(3),\ \delta \in \mathsf{cand}(21)\}$*. Again it is easy to check that* $\mathsf{cand}(21) = \{21\}$ *so we get the pattern* 3421*. Finally, when* $i = 2$ *we get the set* $\{\gamma 4\delta \colon \gamma \in \mathsf{cand}(32),\ \delta \in \mathsf{cand}(1)\}$ *which gives us the pattern* 3241*. In total we have the patterns* 4321,3421*, and* 3241*, which are the underlying classical patterns in Lemma 3.5.*

Before we state the algorithm note that

1. if $u > v$ is an inversion in p' that becomes a non-inversion in $p = S(p')$, then u must stay on the stack until v arrives, and therefore we must shade all the boxes above u and between u and v. Thus there can be no elements of p' in this shaded region. This is handled by line 3 in Algorithm 1;
2. if $u > v$ is an inversion in p' that becomes an inversion in $p = S(p')$, then there must be another element $c > u$ that pops u before v is pushed onto the stack, thus maintaining the inversion. If such an element is present in p' we need not do anything. If there is no such element we need to mark the region above u and between u and v with a "1". This is handled by lines 4–16 in Algorithm 1.

Algorithm 1 $\mathsf{ShadeAndMark}$

Input: $\lambda \in \mathsf{cand}(p)$

1: $n := |p|$
2: marks $:= \emptyset$
3: shades $:= \bigcup \left\{ [\![\lambda^{-1}(v), \lambda^{-1}(u) - 1]\!] \times [\![v, n]\!] \ : \ (u, v) \in \mathsf{ninv}(p) \right\}$
4: **for** $(u, v) \in \mathsf{inv}(p)$ **do**
5:     $i := \lambda^{-1}(u)$
6:     $j := \lambda^{-1}(v)$
7:     **if** $\lambda(\ell) < u$ for all $\ell \in [\![i + 1, j]\!]$ **then**
8:         $M := [\![i, j - 1]\!] \times [\![u, n]\!] \setminus$ shades
9:         **if** $M = \emptyset$ **then**
10:             **return** This candidate cannot be marked properly
11:         **end if**
12:         **if** $m \not\subseteq M$ for all $m \in$ marks **then**
13:             add M to marks and remove all supersets of M
14:         **end if**
15:     **end if**
16: **end for**
17: **return** $(\lambda, \text{shades}, \text{marks})$

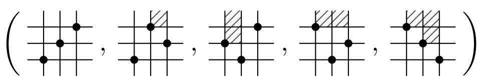
Here $\mathsf{ninv}(p)$ is the set of non-inversions in p, and $\mathsf{inv}(p)$ is the set of inversions in p.

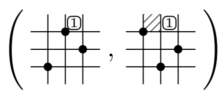
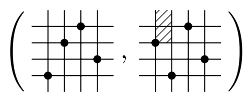
Example 4.3 *For the pattern* $p = 3241$ *in Lemma 3.5 we have* $\mathsf{ninv}(p) = \{(3, 4), (2, 4)\}$ *and* $\mathsf{inv}(p) = \{(3, 2), (3, 1), (2, 1), (4, 1)\}$*. We saw in Example 4.2 that there are three candidates. We only consider* $\lambda = 4321$*, for which* $\lambda^{-1} = \lambda$*. The shading is the union of the sets* $\{1\} \times \{4\}$ *and* $[\![1, 2]\!] \times \{4\}$ *which is* $\{(1, 4), (2, 4)\}$*. The for-loop in the algorithm proceeds as follows.*

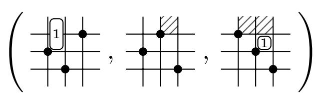
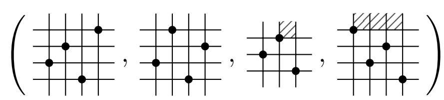
- $(u, v) = (3, 2)$*:* $M = \{(2, 3)\}$ *is added to marks,*
- $(u, v) = (3, 1)$*:* $M = \{(2, 3), (3, 3), (3, 4)\}$ *is a superset of* $\{(2, 3)\}$*, so it is not added to marks,*
- $(u, v) = (2, 1)$*:* $M = \{(3, 2), (3, 3), (3, 4)\}$ *is added to marks,*
- $(u, v) = (4, 1)$*:* $M = \{(3, 4)\}$ *is added to marks and the superset* $\{(3, 2), (3, 3), (3, 4)\}$ *is removed.*

This leaves us with the marking $\{\{(2, 3)\}, \{(3, 4)\}\}$ *which is consistent with Lemma 3.5.*

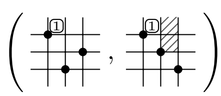
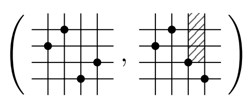
The full algorithm calls ShadeAndMark for all classical patterns in $\mathsf{cand}(p)$. Below is the result of applying this algorithm to all classical patterns of length 3.

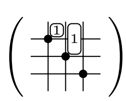
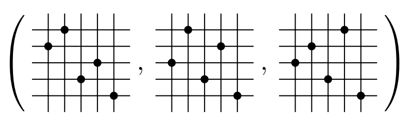
$S^{-1}(\mathrm{Av}(123)) = \mathrm{Av}$ 

$S^{-1}(\mathrm{Av}(132)) = \mathrm{Av}$  $= \mathrm{Av}$ 

$S^{-1}(\mathrm{Av}(213)) = \mathrm{Av}$  $= \mathrm{Av}$ 

$S^{-1}(\mathrm{Av}(231)) =$ See Theorem 3.2

$S^{-1}(\mathrm{Av}(312)) = \mathrm{Av}$  $= \mathrm{Av}$ 

$S^{-1}(\mathrm{Av}(321)) = \mathrm{Av}$  $= \mathrm{Av}$ 

Bouvel and Guibert [BG11] have found a bijection between $S^{-1}(\mathrm{Av}(312))$ and the set of Baxter permutations; they have also found a bijection between $S^{-1}(\mathrm{Av}(231))$ and $S^{-1}(\mathrm{Av}(132))$.

We note that the algorithm can easily be extended to accept a finite list of classical patterns. We also note that a slight modification of algorithm ShadeAndMark has been shown to work with the bubble-sort operator [CU11]; this algorithm can for example be used to prove Proposition 3.3.

5 Open problems

The algorithm above describes the preimage of any set $\mathrm{Av}(p)$ where p is a classical pattern. Can the algorithm be extended to cover the case where p is a mesh pattern, or even a decorated pattern? Solving this problem would automate the description of West-3-stack-sortable permutations. More generally, is there a pattern definition that is stable under S^{-1}?

West [Wes90] conjectured, and Zeilberger [Zei92] proved, that there are $2(3n)!/((n + 1)!(2n + 1)!)$ West-2-stack-sortable permutations. Later Dulucq, Gire and West [DGW96] found these permutations to be in bijection with rooted non-separable planar maps. The enumeration of West-3-stack-sortable permutations is completely open, but knowing the patterns in Theorem 3.9 could provide some insight.

Acknowledgements

We were supported by grant no. 090038013 from the Icelandic Research Fund. We would like to thank the anonymous referees for detailed and constructive comments. The first author also wishes the express his gratitude to Michael Albert, Mike Atkinson, Mathilde Bouvel and Mark Dukes for many interesting and valuable discussions on the topic of sorting operators.

References

[AAB+11] M. H. Albert, M. D. Atkinson, M. Bouvel, A. Claesson, and M. Dukes. On the inverse image of pattern classes under bubble sort. *Journal of Combinatorics*, 2:231–243, 2011.

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

[BG11] M. Bouvel and O. Guibert. Enumeration of permutations sorted with two passes in a stack and D_8 symmetries. Work in progress, 2011.

[Bón03] M. Bóna. A survey of stack-sorting disciplines. *Electron. J. Combin.*, 9(2):Article 1, 16, 2002/03. Permutation patterns (Otago, 2003).

[CU11] A. Claesson and H. Úlfarsson. Preimages of pattern classes under bubble and stack sort. In preparation, 2011.

[DGW96] G. Dulucq, S. Gire, and J. West. Permutations with forbidden subsequences and nonseparable planar maps. *Disc. Math.*, 153(1-3):85–103, 1996.

[HJS+11] Í. Hilmarsson, I. Jónsdóttir, S. Sigurdardóttir, S. Vidarsdóttir, and H. Úlfarsson. Wilf-classification of mesh patterns of short length. In preparation, 2011.

[Knu75] D. E. Knuth. *The art of computer programming*. Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, second edition, 1975. Volume 1: Fundamental algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[Ú11a] H. Úlfarsson. Describing West-3-stack-sortable permutations with permutation patterns. arXiv:1110.1219v2 [math.CO], 2011.

[Ú11b] H. Úlfarsson. A unification of permutation patterns related to Schubert varieties. *Pure Math. Appl. to appear*, 2011.

[Wes90] J. West. *Permutations with forbidden subsequences and stack-sortable permutations*. PhD thesis, MIT, 1990.

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