The Insertion Encoding of Permutations

Michael H. Albert
Department of Computer Science,
University of Otago,
Dunedin, New Zealand
michael.albert@cs.otago.ac.nz

Steve Linton
School of Computer Science,
University of St. Andrews,
St. Andrews, United Kingdom
sal@dcs.st-and.ac.uk

Nik Ruškuc
School of Mathematics and Statistics,
University of St. Andrews,
St. Andrews, United Kingdom
nik@mcs.st-and.ac.uk

Submitted: Aug 29, 2005; Accepted: Sep 14, 2005; Published: Sep 19, 2005
Mathematics Subject Classifications: 05A05, 05A15, 68Q45

Abstract

We introduce the *insertion encoding*, an encoding of finite permutations. Classes of permutations whose insertion encodings form a regular language are characterized. Some necessary conditions are provided for a class of permutations to have insertion encodings that form a context free language. Applications of the insertion encoding to the evaluation of generating functions for classes of permutations, construction of polynomial time algorithms for enumerating such classes, and the illustration of bijective equivalence between classes are demonstrated.

1 Introduction

This paper deals with the problem of describing or enumerating certain collections of permutations. The initial impetus to this investigation was an attempt to understand and unify numerous results concerning the enumeration of *pattern classes of permutations* (defined below), in particular related to the form of their generating functions, polynomial time enumeration algorithms, and coincidences of the enumeration sequences. The book, *Combinatorics of Permutations* by Miklos Bóna [7] provides a useful introduction to this area, and Volume 9(2) of the *Electronic Journal of Combinatorics* contains a wide variety of research papers from this field.

The method we apply is to introduce a uniform encoding of permutations, called the *insertion encoding*. This allows an identification between sets of permutations and words in the language of this encoding.

We now fix some notation and definitions. Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, 2, \ldots, n\}$ and $\mathcal{S} = \cup_{n=0}^{\infty} \mathcal{S}_n$. We denote a permutation $\pi \in \mathcal{S}_n$ by its sequence of values $\pi_1 \pi_2 \cdots \pi_n$.

Definition 1 *Let $\sigma \in S_k$ and $\pi \in S_n$ be given. Then σ* is involved *in π ($\sigma \preceq \pi$) if for some $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, and all $1 \leq s, t \leq k$, $\sigma_s < \sigma_t$ if and only if $\pi_{i_s} < \pi_{i_t}$.*

If σ is not involved in π we say that π *avoids* σ. Slightly more generally the definition of involvement makes sense when π is any sequence of distinct values from some linearly ordered set. Given such a sequence of length n there will be a unique element $\sigma \in S_n$ involved in it – we refer to σ as the *pattern* of π. We say that two sequences are *order isomorphic* if they have the same pattern. Note that all of these definitions can be extended to allow for the presence of duplicated elements in the obvious way, but we do not require such an extension for our discussions.

The definition of involvement may appear somewhat arbitrary. The following discussion illustrates that it is really a natural substructure relationship.

Consider a finite set X equipped with two linear orderings $<$ and $\ll$. Label the elements of X by $\{1, 2, \ldots, n\}$ so that

$$1 < 2 < 3 < \cdots < n$$

Now for each $1 \leq i \leq n$ the element i has some rank π_i in the linear order $\ll$. Clearly $\pi = \pi_1 \pi_2 \cdots \pi_n$ is a permutation, and two structures of this type are isomorphic if and only if the corresponding permutations are the same. Thus the isomorphism classes of such relational structures are in one to one correspondence with permutations. Suppose that $Y \subseteq X$ is a substructure of X, with elements $\{i_1, i_2, \ldots, i_k\}$ where $i_1 < i_2 < \cdots < i_k$ and associated permutation σ. Then

$$\sigma_s < \sigma_t \iff \pi_{i_s} < \pi_{i_t}$$

so σ is involved in π. Conversely any subsequence of π (and thereby any involvement of some permutation in π) determines a substructure of X in this way.

Definition 2 *A* pattern class, *$\mathcal{C}$, is a subset of $\mathcal{S}$ closed downwards under $\preceq$. That is, if $\pi \in \mathcal{C}$ and $\sigma \preceq \pi$ then $\sigma \in \mathcal{C}$. The* spectrum *of a pattern class is the sequence $(|\mathcal{C} \cap S_n|)_{n=1}^{\infty}$.*

Pattern classes are simply ideals of the partially ordered set $(\mathcal{S}, \preceq)$ but, by the preceding remarks are also in one to one correspondence with isomorphism and substructure closed classes of finite sets carrying two linear orders.

It appears to be folkloric that the partially ordered set $(\mathcal{S}, \preceq)$ has precisely eight automorphisms. These form a dihedral group generated by the elements whose actions on structures $(X, <, \ll)$ are given by:

$$\begin{aligned} (X, <, \ll) &\rightarrow (X, >, \ll) \\ (X, <, \ll) &\rightarrow (X, \ll, <). \end{aligned}$$

On permutations of length n these maps correspond to reversal and inversion. In order to verify that every automorphism of $(\mathcal{S}, \preceq)$ belongs to the group generated by these elements it suffices to check that any such automorphism must agree with an element of this group on the structures of size at most four (or equivalently on permutations of length at most four) and that only the identity fixes all structures of size at most four. The images of a pattern class under the automorphisms of $(\mathcal{S}, \preceq)$ are called the *isomorphs* of the class. Most of the significant properties of pattern classes are preserved by these automorphisms so we shall freely replace a class by one of its isomorphs when that is convenient.

One way to specify a pattern class is to provide a set of forbidden patterns.

Definition 3 *Let $B \subseteq \mathcal{S}$. The class of permutations* avoiding B *is:*

$$\mathrm{Av}(B) = \{\pi \in \mathcal{S} \,:\, \forall \beta \in B \; \beta \not\preceq \pi\}.$$

In general, any pattern class $\mathcal{C}$ is of the form $\mathrm{Av}(B)$ where B is the set of $\preceq$ minimal elements of $\mathcal{S} \setminus \mathcal{C}$. This set is sometimes referred to as the *basis* of $\mathcal{C}$.

The study of pattern classes originated with work of Knuth [14] who investigated the class of permutations that can be generated (or sorted) by a single stack. We will return to this example later. Subsequently the study of pattern classes has become an active subfield of combinatorial research. Generally speaking, the aim of such research is to attempt to develop an understanding of the structure of pattern classes. Much of the work in this area has focused on the specific problem of determining the spectrum of specific pattern classes. Many examples of *Wilfian equivalence*, that is, distinct pattern classes with the same spectrum have been observed. The recent resolution by Marcus and Tardos of the Stanley-Wilf conjecture [16] establishes that for each proper class $\mathcal{C}$:

$$L(\mathcal{C}) := \limsup_{n \to \infty} |\mathcal{C} \cap \mathcal{S}_n|^{1/n} < \infty.$$

The number $L(\mathcal{C})$ is called the *Stanley-Wilf limit* of $\mathcal{C}$. When $\mathcal{C} = \mathrm{Av}(\{\pi\})$ this is usually written $L(\pi)$ and in this case the $\limsup$ is known to be an actual limit by results of Arratia [4].

One successful tool in enumerating pattern classes and finding Wilfian equivalences between them has been the *generating tree* approach introduced to this area by West ([22, 23] and see also [6]). Recently, the ECO method ([5, 11] and references therein) has extended this approach to other combinatorial settings. A related method of creating *enumeration* *schemes* was introduced by Zeilberger in [24] and has been significantly extended and generalized by Vatter [19] and includes a Maple package WilfPlus [20].

In this paper we introduce a further construction or interpretation from this general family which we call the *insertion encoding*. We will demonstrate that it can be used to reproduce results previously obtained using the generating tree approach. However, because it allows us to replace pattern classes by their encodings, we may make use of results and techniques from the combinatorics of words and languages. In particular:

- Many classes of interest correspond to regular or context free languages under the insertion encoding which gives a great deal of information about their spectra.
- Bijective correspondences leading to Wilfian equivalences between some pattern classes become transparent as they often simply involve direct transcription of corresponding encodings.
- It is often possible to provide a polynomial time algorithm for the enumeration of a pattern class even when no information about the algebraic properties of the generating function for the class is available.
- The insertion encoding can be used to study the properties of "generalized pattern classes".

In other words, the insertion encoding allows us to discover much of the structure of whole collections of pattern classes at once, while most previous methods required specific tailoring for each individual class.

2 Background and definitions

Every permutation can be generated from the empty permutation by successive insertion of a new maximum element. We call this procedure the *evolution* of a permutation. This observation is used as a basis for the generating tree methodology. In that methodology, a partially constructed permutation belonging to some specified pattern class is viewed as having a number of active sites – points at which a new maximum *could* be inserted while remaining in the pattern class. We modify this viewpoint slightly in that for us an active site in the construction of a permutation is one in which a new maximum element *will* eventually be inserted.

Consider for example the permutation 423615. We trace its evolution as follows:

$$\begin{gathered}\diamond\\ \diamond\,1\,\diamond\\ \diamond\,2\,\diamond\,1\,\diamond\\ \diamond\,23\,\diamond\,1\,\diamond\\ 423\,\diamond\,1\,\diamond\\ 423\,\diamond\,15\\ 423615\end{gathered}$$

We refer to the unfilled parts of the permutation marked by $\diamond$ as *slots*, and a string such as $\diamond 23 \diamond 1 \diamond$ as a *configuration*. The construction terminates when all the slots have been filled. That is, a slot is not just a site which might be occupied, it is one which must be occupied.

This is the main distinction between the insertion encoding and the generating tree approach. As previously remarked, a slot for the insertion encoding corresponds roughly to what is called an active site in the generating tree. However, in the generating tree there is no guarantee that an active site will ever be occupied, and indeed it may become inactive owing to further insertions. The methodology of Vatter's enumeration schemes ([19]) introduces *gap vectors* which again correspond roughly to slots. A gap vector contains more information than a slot configuration does in that it specifies exactly how many elements will be inserted into a particular gap.

At each preliminary stage we index the slots which are present from left to right beginning with 1. Then each insertion is determined by the following information:

- The index of the slot in which it occurs.
- The way in which the slot is affected by the insertion.

There are four different types of insertion of a new maximum element m within a slot:

$$\begin{array}{cl} \diamond \to \diamond m \diamond & \text{represented by } \mathbf{m} \text{ (for middle)}\\ \diamond \to m \diamond & \text{represented by } \mathbf{l} \text{ (for left)}\\ \diamond \to \diamond m & \text{represented by } \mathbf{r} \text{ (for right)}\\ \diamond \to m & \text{represented by } \mathbf{f} \text{ (for fill)} \end{array}$$

If we subscript the insertion type with the slot on which it operates, we obtain a uniquely defined encoding of any permutation. Returning to 423615, its encoding is $\mathbf{m}_1\mathbf{m}_1\mathbf{l}_2\mathbf{f}_1\mathbf{f}_2\mathbf{f}_1$. This encoding is the *insertion encoding*. The choice to base the insertion encoding on maximum elements is an entirely arbitrary one. We could equally well have chosen to insert new minimum elements, or to insert elements from left to right, or right to left (in the latter two cases the slots would represent gaps in the currently constructed sequence of values rather than positions). However, these choices simply reflect certain basic symmetries of the partially ordered set $(S, \preceq)$ and the choice to insert maximum values corresponds most naturally with the existing literature on the generating tree approach. The alphabet of the insertion encoding is denoted Λ, and for a positive integer k, $\Lambda_k \subseteq \Lambda$ consists of the symbols whose subscript is at most k.

There is an infinite automaton, which we shall call PermAut, which accepts precisely the words representing the encodings of permutations. It has states indexed by the number of free slots. The initial state is state 1, and the final state is state 0. From state $k \geq 1$ we have the following transitions:

$$\begin{array}{ll} \mathbf{m}_i & \text{for } 1 \leq i \leq k, \text{ to state } k+1 \\ \mathbf{l}_i & \text{for } 1 \leq i \leq k, \text{ to state } k \\ \mathbf{r}_i & \text{for } 1 \leq i \leq k, \text{ to state } k \\ \mathbf{f}_i & \text{for } 1 \leq i \leq k, \text{ to state } k-1 \end{array}$$

Most applications of the insertion encoding use, at least implicitly, a variation of PermAut in one of two ways. Either the number of states is constrained, leading to regular classes discussed in Section 4, or the indexes of the states are viewed as the size of a stack and the language is constrained to a finite alphabet in some fashion leading to context free classes discussed in Section 5.

Since PermAut accepts exactly the insertion encodings of permutations, the generating function of its language and that of permutations must agree. Using the methods of [12] this generating function is easily represented as a continued fraction and we thereby obtain the (formal) identity:

$$\sum_{i=1}^{\infty} i!t^i = \cfrac{t}{1-2t-\cfrac{(2\times 1)t^2}{1-4t-\cfrac{(3\times 2)t^2}{1-6t-\cfrac{(4\times 3)t^2}{1-8t-\cfrac{(5\times 4)t^2}{1-10t-\cdots}}}}}$$

For clarity, we sometimes use negative subscripts in counting slots from the right. That is $\mathbf{m}_{-1}$ represents a middle insertion into the rightmost slot and $\mathbf{r}_{-2}$ represents an insertion on the right hand end of the second slot from the right.

There are also situations where a modification of the insertion encoding is desirable. Suppose that we are interested in studying a collection of permutations $\mathcal{C}$ with the property that for any $\pi, \tau \in \mathcal{C}$ the permutation $\pi \oplus \tau$ which has an initial segment of small values order isomorphic to π followed by a segment of larger values order isomorphic to τ is also in $\mathcal{C}$ (for example $21 \oplus 312 = 21534$). Then it is natural to retain a "free slot" on the right hand end of any configuration. We enforce this by simply forbidding the $\mathbf{f}$ and $\mathbf{r}$ operations in the final slot. Of course there is a dual situation where it is desirable to leave a free slot at the left hand end of every configuration.

We begin with a pair of examples which will illustrate the simplicity and transparency of the insertion encoding.

3 The classes Av(312) and Av(321)

The class Av(312) is the collection of permutations that can be produced from initial input in order $123\cdots n$ by passing it through a single stack. This is the class (or a variation of it) originally discussed by Knuth. The operation sequence of a stack when producing a permutation in Av(312) is uniquely determined, and this observation establishes that Av(312) is enumerated by the Catalan numbers. The class Av(321) is the collection of permutations that can be produced from initial input in order $123\cdots n$ by passing it through a pair of parallel queues. This class is also enumerated by the Catalan numbers and a variety of bijections between it and Av(312) are known. Both these classes have straightforward encodings as context free languages via the insertion encoding, and moreover this encoding immediately makes their Wilf-equivalence clear.

Proposition 4 *The permutations in* Av(312) *are precisely those whose insertion encoding uses only the symbols* $\mathbf{f}_1$, $\mathbf{l}_1$, $\mathbf{r}_1$ *and* $\mathbf{m}_1$.

Proof: Consider a state of the insertion encoding when two or more slots are available. If an element is inserted into any slot other than the first one, then it, together with any element belonging to the left boundary of the slot into which it was inserted make a 12 pair. When, at some later point, an element is added to the first slot (as it must be) this will create a 312 pattern. Thus, in order to avoid creating a 312 pattern, only the symbols $\mathbf{f}_1$, $\mathbf{l}_1$, $\mathbf{r}_1$ and $\mathbf{m}_1$ may be used.

Conversely, if the encoding of π uses only these symbols then, whenever an element is inserted, all slots to its left will be filled before any slots to its right are. Thus no element can be the 1 of a 312 pattern, and so no 312 patterns can occur. ☐

Thus we have provided an encoding of the elements of Av(312) over a four element alphabet. Moreover, this encoding can be decoded in an online fashion. That is, if the first k letters of the insertion encoding of some permutation are known, then the relative positional order of the smallest k values within the permutation is also known. After suppressing the superfluous subscript, the unambiguous grammar for this language is:

$$S \to \mathbf{f} \mid \mathbf{l}S \mid \mathbf{r}S \mid \mathbf{m}SS.$$

Using the observations of Chomsky and Schutzenberger [8] the generating function, $f(t)$, for the spectrum of Av(312) is seen to satisfy:

$$f = t + 2tf + tf^2.$$

Extra information about the class can be derived from the grammar above. To give just a single simple example, suppose that we wish to compute its generating function $g(z,t)$ where the coefficient of $z^k t^n$ is the number of permutations of length n in the class having precisely k descents. We observe that the $\mathbf{m}$ and $\mathbf{r}$ letters insert a symbol which will be the lower end of a descent, while the $\mathbf{f}$ and $\mathbf{l}$ symbols never do this. We may then read directly from the grammar that:

$$g = t + tg + ztg + ztg^2.$$

The question of whether to include the empty permutation within a pattern class, or rather, whether the generating function of a pattern class should have a non-zero constant term is one which does not seem to have a uniformly correct answer. Generally speaking, when using the standard insertion encoding it is usually more convenient not to include a constant term, as the final state with no slots free differs from the initial state. If the modified encoding is used where a free slot is always maintained then it is usually more convenient to include a constant term.

We turn now to Av(321). We use the modification of the insertion encoding, where additions at the right hand end of the final slot are forbidden. In this form, the empty permutation is included as an element of every class. We also adopt the convention that in this modified form the only letters which may occur when only one slot is free are $\mathbf{m}_{-1}$ and $\mathbf{l}_{-1}$. This convention is useful in reminding us of the special rôle of this slot.

Proposition 5 *The following grammar describes the encodings of the class* Av(321) *in the modified form presented above:*

$$\begin{array}{rcl} S & \to & \epsilon \,|\, \mathbf{l}_{-1}S \,|\, \mathbf{m}_{-1}TS \\ T & \to & \mathbf{f}_1 \,|\, \mathbf{l}_1 T \,|\, \mathbf{l}_{-1}T \,|\, \mathbf{m}_{-1}TT. \end{array}$$

Proof: The symbol S should be interpreted as representing a word which represents an element of the class, while T represents a word that removes a slot from a configuration containing two or more slots. It is easy to see that all words in the language represented by this grammar are the encodings of permutations (in the modified insertion encoding).

If a permutation π avoids 321 then, in constructing π, no symbol can ever be inserted into a slot other than the first or the last. Otherwise, together with the symbol in the right boundary of its slot, it would form a 21 which would subsequently become a 321 when some earlier slot was filled. Similarly any occurrence of the letters $\mathbf{m}_1$ and $\mathbf{r}_1$ (which in the modified form are used only when two or more slots are present) would eventually lead to a 321. So all the encodings of permutations avoiding 321 are in the language represented by this grammar.

Conversely, suppose that a 321 does occur in a permutation π, say $\pi = \cdots c \cdots b \cdots a \cdots$ with $c > b > a$. Consider the insertion encoding at the moment when the symbol b is inserted. As a is already present, b is not being inserted into the rightmost slot. As c will eventually be added to its left it is not being inserted with the symbols $\mathbf{l}_1$ or $\mathbf{f}_1$. So the encoding of π is not in the language represented by this grammar. $\square$

Now to see the bijective correspondence between Av(312) and Av(321) we need only rewrite the grammar for the language of the encodings of the former class into the modified form. This gives:

$$\begin{array}{rcl} S' & \to & \epsilon \,|\, \mathbf{l}_{-1} S' \,|\, \mathbf{m}_{-1} T' S' \\ T' & \to & \mathbf{f}_1 \,|\, \mathbf{l}_1 T' \,|\, \mathbf{r}_1 T' \,|\, \mathbf{m}_1 T' T'. \end{array}$$

which can be obtained from the above by substituting $\mathbf{r}_1$ for $\mathbf{l}_{-1}$ and $\mathbf{m}_1$ for $\mathbf{m}_{-1}$ in any production of the second type.

4 Regular classes

Our first family of applications of the insertion encoding concerns pattern classes whose insertion encodings form regular languages. Such *regular pattern classes* have a variety of useful properties, including the rationality of their generating functions, and simple algorithms for recognizing elements of a class or for constructing the basis elements. If $\mathcal{A}$ is a regular pattern class we will freely refer to the (or a) $\mathcal{A}$-automaton, which is some finite state automaton that recognizes the insertion encodings of elements of $\mathcal{A}$. We may use deterministic or non-deterministic automata as circumstances dictate, and refer to transitions between states that consume no input as ϵ-transitions. The transition function for an $\mathcal{A}$-automaton will be denoted $\tau_{\mathcal{A}}$.

For any regular language, L, there exists a fixed integer k such that if p is a prefix of some word in L, then there is a word pw in L with $|w| \leq k$. The minimum number of symbols required to complete an insertion encoding from a configuration including s slots is s. Therefore, if the language of a pattern class is regular there must be some uniform bound on the number of slots that can occur in any configuration leading to an element of the class.

This bound serves to restrict the language of the insertion encoding for a regular pattern class to a finite subset of Λ. A priori we might have been willing to allow the definition of a regular pattern class to include some translation mechanism from its finite alphabet to Λ. However, the previous remark shows that this is not necessary.

Observe also that if $\mathcal{A}$ is a regular pattern class, then for each state q of an $\mathcal{A}$-automaton, there is some integer $s(q)$ such that there is a path from q to the final state which consumes $s(q)$ input elements, but no such path which consumes less input. Thus $s(q)$ represents the number of slots present in some and hence *any* configuration whose creation leads to state q.

Definition 6 *For each positive integer k the set $\mathcal{SB}(k)$ of permutations whose insertion encoding never includes more than k slots is called the set of k* slot bounded permutations.

Proposition 7 *For each positive integer k the set $\mathcal{SB}(k)$ is a pattern class. Its basis consists of the $(k+1)!k!$ permutations in $\mathcal{S}_{2k+1}$ of the form babab $\cdots$ bab where the positions marked by b's are occupied by the numbers $\{k+1, k+2, \ldots, 2k+1\}$ while those marked by a's are occupied by the numbers $\{1, 2, \ldots, k\}$.*

Proof: Each of the specified basis elements has an insertion encoding which passes through a configuration with $(k + 1)$ slots precisely at the point where all the elements marked a have been added. So, none of these permutations belong to $\mathcal{SB}(k)$.

Now consider any permutation $\pi \notin \mathcal{SB}(k)$. The insertion encoding of π passes through configurations with more than k slots. Since the number of slots changes by at most one with each symbol of the encoding, there exists such a configuration with exactly $k + 1$ slots. Take any k elements of π, one from each block of elements between successive slots of this configuration. Take a further $k+1$ elements of π so that one of them is placed into each of the slots of this configuration. These $2k + 1$ elements witness the involvement in π of one of the permutations in $\mathcal{S}_{2k+1}$ of the form $babab \cdots bab$.

Thus $\mathcal{SB}(k)$ is the set of permutations avoiding these elements, so it is a pattern class, and as these elements are all of the same length, hence form an antichain with respect to involvement, they are its basis. □

It is immediate that the language representing the insertion encodings of the elements of $\mathcal{SB}(k)$ is regular since it is the language accepted by PermAut, with all the states of index $k + 1$ or higher deleted. The generating functions for these classes are easily obtained by curtailing the continued fraction representation of the generating function for the language accepted by PermAut. In particular, the generating function for the class $\mathcal{SB}(1)$ is $t/(1 - t)$ and those for $\mathcal{SB}(2)$ and $\mathcal{SB}(3)$ are

$$\frac{t - 4t^2}{1 - 6t + 6t^2} \quad \text{and} \quad \frac{t - 10t^2 + 12t^3}{1 - 12t - 30t^2 + 12t^3}.$$

The observation preceding the definition of $\mathcal{SB}(k)$ shows that any pattern class whose language with respect to the insertion encoding is regular must be a subclass of $\mathcal{SB}(k)$ for some k. Given such a class and a description of its language (or of the automaton accepting it) we would have immediate access to its spectrum (via the transfer matrix approach), knowledge of properties of its generating function (it is rational) and a linear time recognition algorithm for the class (just run the automaton). So it would be useful to have a large supply of such classes. Such a supply is provided by the following theorem.

Theorem 8 *Let $\mathcal{C}$ be a permutation class that is a subclass of $\mathcal{SB}(k)$ for some k. The following are equivalent:*

- *The language of $\mathcal{C}$ is regular.*
- *There is a regular language defining a subset of $B \subseteq \mathcal{SB}(k)$ such that $\mathcal{C} = \mathrm{Av}(B) \cap \mathcal{SB}(k)$.*

In fact B can, but need not, be chosen to consist of those elements of the basis of $\mathcal{C}$ which belong to $\mathcal{SB}(k)$, and there is an effective procedure for passing from the language of B to that of $\mathcal{C}$ and vice versa.

The form of this theorem is very similar to that of Theorem 2 in Section 2 of [2]. Its proof is identical to the proof of that theorem, and depends only on the following modification of Proposition 5 from the same section of that paper.

Lemma 9 *For each positive integer k there is a non-deterministic transducer* $\textsc{Del}_k$ *with the property that for all $\pi \in \mathcal{SB}(k)$, the words produced by the operation of* $\textsc{Del}_k$ *when applied to the insertion encoding of π are precisely the insertion encodings of all the permutations σ with $\sigma \preceq \pi$.*

The proof of this lemma consists of an explicit, and uninteresting, description of the transducer $\textsc{Del}_k$. We therefore defer it to the end of this section in order to discuss some of the consequences of Theorem 8 as well as some constructions which preserve regularity of pattern classes in greater detail.

Corollary 10 *Any finitely based subclass of $\mathcal{SB}(k)$ is encoded by a regular language in the insertion encoding and therefore has a rational generating function. Given the basis, there is an effective procedure which produces a linear time recognition algorithm for the class.*

In order to describe how to determine whether a class is a subclass of $\mathcal{SB}(k)$ for some k and also to prepare for some closure results for regular classes we need to introduce some structural operations on permutations and classes.

Definition 11 *Let $\pi \in S_n$ and $\tau \in S_k$ be given. A* horizontal juxtaposition *of π with τ is any permutation $\sigma \in S_{n+k}$ such that the pattern of the leftmost n values of σ is the same as the pattern of π, and that of the rightmost k values is the same as the pattern of τ. That is, for $1 \leq i, j, \leq n$ $\sigma_i < \sigma_j$ if and only if $\pi_i < \pi_j$ while for $1 \leq i, j \leq k$ $\sigma_{n+i} < \sigma_{n+j}$ if and only if $\tau_i < \tau_j$. A* vertical juxtapostion *of π with τ is any permutation $\sigma \in S_{n+k}$ such that the pattern of the smallest n values of σ is the same as the pattern of π, and that of the largest k values is the same as the pattern of τ. The* direct sum *$\pi \oplus \sigma$ is that horizontal juxtaposition of π with σ in which the leftmost n values are all smaller than the remaining k values and the direct difference $\pi \ominus \sigma$ is that horizontal juxtaposition of π with σ in which the leftmost n values are all larger than the remaining k values.*

Example 12 *If $\pi = 123$ and $\sigma = 21$ then 15243 is a vertical juxtaposition of π with σ, 13452 is a horizontal juxtaposition, the direct sum is 12354 and the direct difference is 34521.*

For pattern classes $\mathcal{A}$ and $\mathcal{B}$ we define the vertical juxtaposition (horizontal juxtaposition etc.) of $\mathcal{A}$ with $\mathcal{B}$ to consist of all vertical juxtapositions of some $\alpha \in \mathcal{A}$ with some $\beta \in \mathcal{B}$. We also define the *sum closure* of $\mathcal{A}$ to be the set of all permutations of the form $\alpha_1 \oplus \alpha_2 \oplus \cdots \oplus \alpha_k$ for some sequence of permutations $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathcal{A}$. The difference closure is defined similarly. It is easy to see that all of these sets are in fact pattern classes.

Proposition 13 *A class $\mathcal{C}$ is a subclass of $\mathcal{SB}(k)$ for some k if and only if there are permutations $\pi_{II}, \pi_{ID}, \pi_{DI}, \pi_{DD} \in \mathcal{S} \setminus \mathcal{C}$ where π_{II} is a vertical juxtaposition of two increasing permutations, π_{ID} a vertical juxtaposition of an increasing and a decreasing permutation, π_{DI} a vertical juxtaposition of a decreasing and an increasing permutation and π_{DD} a vertical juxtaposition of two decreasing permutations.*

Proof: Suppose first that $\mathcal{C} \subseteq \mathcal{SB}(k)$. Then the four requisite permutations can easily be found among the basis elements of $\mathcal{SB}(k)$ by choosing the small and large elements (respectively a's and b's in the notation of Proposition 7) to be monotone sequences of either type (increasing or decreasing).

Conversely, suppose that four permutations of the type described can be found in $\mathcal{S} \setminus \mathcal{C}$. Without loss of generality we may choose each of these four permutations of length $2n$ for some fixed n and to have the pattern $abab \cdots ab$ where again a denotes a "small" element and b a "large" one. For instance if $\pi_{II} = 314256 \in \mathcal{S} \setminus \mathcal{C}$ we could equally well take $\pi_{II} = 15263748$ which involves the original choice as the subsequence 526378. Let $k = n^4$ and let θ be any basis element of $\mathcal{SB}(k)$. By the Erdös-Szekeres theorem, among the k smallest elements of θ there is a subsequence of length n^2 which is monotone. Choose such a subsequence, and take n^2 large elements from θ so the pattern they form with the chosen subsequence is $aba \cdots ab$. Now, among these large elements there is a subsequence of length n which is also monotone. Choose such a subsequence and interpose small elements from the original monotone subsequence to produce a subsequence of length $2n$ of the form $abab \cdots ab$ within θ in which the a's and the b's form monotone sequences. This subsequence witnesses the involvement of one of the four π's in θ. As θ was arbitrary, none of the basis elements of $\mathcal{SB}(k)$ belong to $\mathcal{C}$ and hence $\mathcal{C} \subseteq \mathcal{SB}(k)$ as required. $\square$

This result establishes that it is decidable whether a finitely based pattern class is contained in $\mathcal{SB}(k)$ for some k. Together with Theorem 8 it generalizes a result of Vatter [18] which used a careful analysis of generating trees to establish that any finitely based class whose basis includes a permutation obtained from an increasing sequence by inserting a new maximum element, and a permutation obtained from a decreasing sequence by inserting a new maximum element has a rational generating function. In this case the upper part of the juxtapositions required can be taken to be a singleton.

Proposition 14 *Let $\mathcal{A}$ and $\mathcal{B}$ be classes which are regular for the insertion encoding. Then the direct sum, direct difference and horizontal juxtaposition of $\mathcal{A}$ with $\mathcal{B}$ are regular for the insertion encoding as are the sum and difference closures of $\mathcal{A}$. The vertical juxtaposition of $\mathcal{A}$ and $\mathcal{B}$ is regular if and only if one of the two classes is finite.*

Proof: First consider the direct sum $\mathcal{A} \oplus \mathcal{B}$. Take a finite state automaton which recognizes the insertion encodings of elements of $\mathcal{A}$. From each final state (which we still keep as an accepting state) add an ϵ transition to the initial state of an automaton which recognizes the insertion encodings of elements of $\mathcal{B}$. For the sum closure of $\mathcal{A}$ add to the $\mathcal{A}$-automaton an ϵ transition from each of its final states to its initial state. A similar construction proves the regularity of $\mathcal{A} \ominus \mathcal{B}$ and of the difference closure of $\mathcal{A}$.

Let $\mathcal{C}$ be the horizontal juxtaposition of $\mathcal{A}$ and $\mathcal{B}$. Consider the insertion encoding of some element of $\theta \in \mathcal{C}$. Choose $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ such that θ is a horizontal juxtaposition of α with β. This defines a labelling of the symbols of the insertion encoding of θ as $\mathcal{A}$-symbols, and $\mathcal{B}$-symbols, according to whether or not the corresponding element of θ comes from α or β. This labelling must be consistent in that if some symbol is labelled as an $\mathcal{A}$-symbol, then any later symbol representing an element to the left of this one must also be so labelled, and dually for $\mathcal{B}$-symbols. Conversely, given any permutation τ, and a labelling of its insertion encoding as $\mathcal{A}$-symbols and $\mathcal{B}$-symbols satisfying these consistency criteria, if the interpretation of $\mathcal{A}$-symbols defines a permutation in $\mathcal{A}$ and the interpretation of the $\mathcal{B}$-symbols (suitably reindexed) defines a permutation in $\mathcal{B}$ then τ belongs to the horizontal juxtaposition of $\mathcal{A}$ with $\mathcal{B}$.

These observations indicate how to construct an automaton for the horizontal juxtaposition of $\mathcal{A}$ with $\mathcal{B}$ from deterministic ones for $\mathcal{A}$ and $\mathcal{B}$ respectively. Let the transition functions of the $\mathcal{A}$ and $\mathcal{B}$-automata be $\tau_{\mathcal{A}}$ and $\tau_{\mathcal{B}}$ respectively.

In a configuration leading towards the construction of a permutation in $\mathcal{C}$, some initial segment of slots will be associated with $\mathcal{A}$, and some final segment with $\mathcal{B}$. There may be one shared slot between these two segments. The meaning of a shared slot s is that during the subsequent evolution of the permutation at least one child of s will receive an $\mathcal{A}$-symbol and at least one child of s will receive a $\mathcal{B}$-symbol. We take the states of the $\mathcal{C}$ automaton to be triples (q, r, δ) where q is a state of the $\mathcal{A}$-automaton, r a state of the $\mathcal{B}$ automaton and $\delta \in \{0, 1\}$ is the number of shared slots. There is a special initial state that has ϵ transitions to each of: $(q_0, r_0, 1)$, $(q_f, r_0, 0)$ and $(q_0, r_f, 0)$ where q_0 and r_0 are the initial states of the $\mathcal{A}$ and $\mathcal{B}$-automata and q_f and r_f are (arbitrarily chosen) final states.

Suppose that we are in state (q, r, δ) and read the letter p_i where $p \in \{\mathbf{f}, \mathbf{l}, \mathbf{r}, \mathbf{m}\}$. If $1 \leq i \leq s(q) - \delta$ then the resulting state of the $\mathcal{C}$-automaton is

$$(\tau_{\mathcal{A}}(q, p_i), r, \delta).$$

Similarly, if $s(q) + 1 \leq s(q) + s(r) - \delta$ the resulting state of the $\mathcal{C}$-automaton is

$$\tau_{\mathcal{C}}((q, r, \delta), p_i) = (q, \tau_{\mathcal{B}}(r, p_{i-s(q)+\delta}), \delta).$$

This covers all insertions, except those into a shared slot, that is when $i = s(q)$ and $\delta = 1$. By our previous convention, a shared slot may not be filled. Suppose first that $p_i = \mathbf{l}_i$. The element inserted must be an $\mathcal{A}$-symbol (otherwise the shared slot will never receive an $\mathcal{A}$-symbol). There are two possible transitions depending on whether or not the slot is still to be shared, namely to

$$(\tau_{\mathcal{A}}(q, \mathbf{f}_i), r, 0) \text{ or } (\tau_{\mathcal{A}}(q, \mathbf{l}_i), r, 1).$$

Similarly, when $p_i = \mathbf{r}_i$ the resulting state is one of

$$(q, \tau_{\mathcal{B}}(r, \mathbf{f}_1), 0) \text{ or } (q, \tau_{\mathcal{B}}(r, \mathbf{r}_1), 1).$$

Finally, a similar analysis applies when $p_i = \mathbf{m}_i$. This time there are four possible transitions to:

$$(\tau_{\mathcal{A}}(q, \mathbf{r}_i), r, 0),\ (\tau_{\mathcal{A}}(q, \mathbf{m}_i), r, 1),\ (q, \tau_{\mathcal{B}}(r, \mathbf{l}_1), 0), \text{ or } (q, \tau_{\mathcal{B}}(r, \mathbf{m}_1), 1).$$

The final states of the $\mathcal{C}$-automaton are all the states of the form $(q, r, 0)$ where q and r are final states of the $\mathcal{A}$- and $\mathcal{B}$-automata.

As regards the vertical juxtaposition, if neither class is finite then the alternating vertical juxtaposition of two permutations of length n, one from each class, shows that the vertical juxtaposition of the classes is not slot bounded. If either class is finite, it is immediately clear that their juxtaposition will be regular. For example, when $\mathcal{B}$ is finite (the more difficult case), the basic automaton recognizing $\mathcal{A}$ is adjusted to allow the creation of certain slots reserved for the $\mathcal{B}$ elements. It is easiest to simply create automata for recognizing the vertical juxtaposition of $\mathcal{A}$ with a single element β for each $\beta \in \mathcal{B}$ and then appeal to the closure of regular languages under finite union. □

The asymmetry of regularity with respect to horizontal and vertical juxtaposition is a consequence of our (arbitrary) choice that the insertion encoding represents the elements of a permutation taken in value order. Had we instead worked with elements in position order, then vertical juxtapositions would have preserved regularity, and horizontal juxtapositions would tend to destroy it. However, we do feel that this apparent asymmetry does represent a possible flaw of the insertion encoding in general.

All of the examples of regular classes given in [2] are in fact regular with respect to the insertion encoding. On the other hand, the deletion transducer is significantly more complicated for the insertion encoding than for any of these examples, so for practical computational purposes it is still worthwhile to operate within these subclasses if possible. Likewise the caveat noted in that paper, that the results given, while effective are not necessarily practical since they generally involve several iterations of determinization of non-deterministic finite automata, apply even more strongly to the languages and automata associated with the insertion encoding.

Recently, the Stanley-Wilf limit of the class Av(4231) was shown to exceed 9.35 essentially by constructing the automaton for the class $\mathcal{SB}(13) \cap \mathrm{Av}(4231)$ and evaluating the largest eigenvalue of the corresponding transition matrix [3]. The significance of this result is that it refutes a conjecture of Arratia [4] that $L(\pi) \leq (k-1)^2$ for any $\pi \in \mathcal{S}_k$. The qualifier "essentially" is required above because the same modification of the insertion encoding as we used in discussing Av(321) was used to obtain this result. Previous attempts to construct these automata using the general methodology provided by Theorem 8 had failed for seven or more slots owing to the exponential complexity of the underlying algorithms.

Proof: (of Lemma 9) As k is fixed throughout, we will refer simply to $\mathcal{SB}$ and to Del rather than $\mathcal{SB}(k)$ and $\textsc{Del}_k$ throughout the proof. The purpose of Del, thought of as acting on the permutation π rather than on its insertion encoding, is to delete symbols. So to describe the construction of Del we imagine some *target* $\sigma \preceq \pi$, obtained by deleting a fixed set of symbols from π and we arrange that σ (or rather its insertion encoding) is realized by some run of Del on the insertion encoding of π. As the proof proper is quite technical and case ridden, we would like to begin with a few explanatory remarks.

Some parts of DEL are easy to understand. For instance if we wish to delete an element encoded by a symbol $\mathbf{r}_2$ then often we can simply delete that symbol from the encoding as the slot in which it is inserted is unaffected, and will be filled later. On the other hand if we delete an element inserted by $\mathbf{m}_2$ then DEL must "remember" that where originally there were two slots there is now only one, though this may be split into two again by a later $\mathbf{r}_2$ or $\mathbf{l}_3$ either of which would need to be transliterated as $\mathbf{m}_2$. Even more confusingly if we intend to delete a symbol encoded by $\mathbf{f}_3$, then DEL must make a change to the *previous* symbol which affected that slot to ensure that it is closed. This seems to be contrary to the allowed operation of a transducer, except that we should remember that we can view our intended target σ as given, and so we, or rather the transducer, can determine that this modification needs to be made when that previous symbol arrives.

Let us begin with an extended example. Consider the permutation:

$$\mathbf{m}_1\mathbf{m}_1\mathbf{r}_3\mathbf{l}_2\mathbf{f}_2\mathbf{f}_2\mathbf{f}_1 \quad \text{i.e.} \quad 7245163,$$

and also the involved subpermutation 7513 (we dispense with reindexing this as 4312 since relative order is all we care about) whose encoding is $\mathbf{m}_1\mathbf{f}_2\mathbf{r}_1\mathbf{f}_1$. We can follow the evolution of these two permutations in parallel:

$$\begin{array}{cccc}
\diamond 1 \diamond & \mathbf{m}_1 & \mathbf{m}_1 & \diamond 1 \diamond \\
\diamond 2 \diamond 1 \diamond & \mathbf{m}_1 & \epsilon & \diamond 1 \diamond \\
\diamond 2 \diamond 1 \diamond 3 & \mathbf{r}_3 & \mathbf{f}_2 & \diamond 13 \\
\diamond 24 \diamond 1 \diamond 3 & \mathbf{l}_2 & \epsilon & \diamond 13 \\
\diamond 2451 \diamond 3 & \mathbf{f}_2 & \mathbf{r}_1 & \diamond 513 \\
\diamond 245163 & \mathbf{f}_2 & \epsilon & \diamond 513 \\
7245163 & \mathbf{f}_1 & \mathbf{f}_1 & 7513
\end{array}$$

The points worth noting in this example are the following:

- When 3 is added, it fills the slot to the right of 1 rather than being added on its right. This is because we intend to subsequently delete 6 which is the only other element destined for this slot.

- When 5 is added it is added on the right of the first slot rather than filling the second slot as it did in the original permutation. In fact, at this point there is no second slot. This is because the first slot (after the deletion of 2) is really a superposition of the first two slots of the original permutation. So, filling the second half of it doesn't fulfil the obligation also to fill the first half.

We can design DEL to cope with both of these points. The first point should be viewed as DEL amending some $\mathbf{r}$ to an $\mathbf{f}$. This must result in a state where an attempt to add to the slot which the $\mathbf{r}$ would have left available must be replaced by an ϵ (as happens to 6). The second point is dealt with as indicated in its description. That is, slots which were separate initially may be superimposed. That then requires later additions which break the superposition up in some fashion to be changed. In an extreme case we might have three successive slots, say the first three, superimposed. A subsequent $\mathbf{f}_2$ would in fact be translated into an $\mathbf{m}_1$ and the superposition would break down.

The states of DEL can be described as follows. For each $s \leq k$ there is a collection of states identified with sequences $a_1 a_2 \cdots a_s$ of non-negative integers. These sequences satisfy the following restrictions:

- $a_1 \in \{0, 1\}$,
- $a_{i+1} \in \{0, a, a+1\}$ where $a = \max\{a_j : j \leq i\}$.

Note in particular that $0 \leq a_i \leq k$ in any case and so the number of states is finite. We refer to the a_i (or, technically, to the pairs (i, a_i)) as the *components* of the state. States are to be interpreted as strings so that operations such as removal of values, or replacement of values by sequences of values entail a reindexing of components where necessary. When we refer to incrementing or decrementing a value, the amount of the increment/decrement is always 1.

The "intent" of $a_1 a_2 \cdots a_s$ is to represent a state reached when some prefix of the encoding of some $\pi \in \mathcal{SB}$ has been processed where the corresponding configuration for π has s slots available. That is the individual components of the state represent the actual slots which would be present if we were producing π. The values, a_i, at these components represent the actual slots present (and their indices) at this point with respect to the target permutation σ. Some slots are indexed by 0's indicating that any subsequent input symbols which would have occupied them will be deleted, while blocks of equal values among the remainder indicate slots which are superimposed in the deleted version because the input symbols separating them which came from the original permutation have been deleted. Note that a 0 can occur within such a block, as witnessed by the involvement of 534 in 516234.

Now we describe the transitions from state $a_1 a_2 \cdots a_s$. Since the source sequence, that is the insertion encoding of π, has only s slots available, we need only deal with input symbols whose subscripts come from 1 through s. Suppose that the subscript is i. We refer to the component a_i as the current component, and denote its value by a. The output symbols and transitions available differ sharply according to whether or not a is 0.

Case $(a = 0)$. In this case we are obliged to delete this input symbol, that is, output ϵ. So our choices are simple.

$\mathbf{f}_i$ Output ϵ. Move to the state obtained by removing the current component.

$\mathbf{r}_i$ **or** $\mathbf{l}_i$ Output ϵ. Leave the state unchanged.

$\mathbf{m}_i$ Output ϵ. Move to the state obtained by replacing the current component by 00.

Case ($a \neq 0$). This situation is more complicated as there are multiple types of action which we might take. We generally need to make reference to the last preceding non-zero, and first following non-zero values of components within the state. We refer to this information as the *local situation* and denote it by a pair of symbols, either ==, =<, <= or <<. In the cases where there is no preceding non-zero value we use < and likewise when there is no following non-zero value. The notation should be self-explanatory, but just to be clear the situation <= refers to a state in which the last non zero value preceding the current component was smaller than a (or no such value exists) and the next non zero value following the current component is equal to a (and definitely exists).

Now we deal with each type of input symbol in turn. In a few cases we provide a detailed explanation for the transition. Otherwise we leave these explanations to be provided by the reader.

If the input symbol is $\mathbf{f}_i$ it may not be deleted as it eliminates a slot. If we intended to delete it, we should have avoided creating the slot earlier. The output symbol depends on the local situation.

(==)

- Output $\mathbf{m}_a$. Move to the state obtained by incrementing all non-zero a_j for $j > i$ by 1 and then removing the current component.
  Explanation: Filling a central part of a superposition breaks the superposition into distinct states.

(=<)

- Output $\mathbf{r}_a$. Move to the state obtained by removing the current component.
  Explanation: Filling the right side of a superposition, is like adding on the right of the superimposed states.

(<=)

- Output $\mathbf{l}_a$. Move to the state obtained by removing the current component.

(<<)

- Output $\mathbf{f}_a$. Move to the state obtained by decrementing the non-zero a_j's for $j > i$ and then removing the current component from the sequence.

The cases where the input symbol is $\mathbf{r}_i$ or $\mathbf{l}_i$ are similar to one another so we consider only the former. Suppose first that we intend to delete this symbol, i.e. output ϵ. Then simply do so, but make no change to the underlying state. Otherwise, consider again the local situation.

(==)

- Output $\mathbf{m}_a$. Move to the state obtained by incrementing all non-zero a_j for $j > i$ and either leaving the value of the current component unchanged, or change it to 0.
  Explanation: We have broken a superposition, and must also indicate whether an element will be added later into this slot, or whether all such elements are deleted in the target.

$(<=)$

- Output $\mathbf{m}_a$. Move to the state obtained by incrementing all non-zero a_j for $j > i$ and leaving a_i unchanged. Or, output $\mathbf{l}_a$ and move to the state obtained by changing the value of the current component to 0.
  Explanation: Either we break a superposition, or we intend to delete all further elements of the superposition to the left of this one.

$(=<)$

- Output $\mathbf{r}_a$. Leave the state unchanged, or change the value of the current component to 0.

$(<<)$

- Output $\mathbf{r}_a$ and leave the state unchanged. Or, output $\mathbf{f}_a$, and move to the state obtained by changing the value of the current component to 0 and decrementing all subsequent non-zero values.

Finally, consider the input symbol $\mathbf{m}_i$. Suppose first that we decide to delete this symbol. Then we output ϵ. We obtain our new state from the original state by replacing the current component by any one of aa, $a0$, or $0a$ (but not 00) depending on our subsequent intentions. If we do not wish to delete the input symbol then consider the local situation again.

$(==)$

- Output $\mathbf{m}_a$. Move to a state obtained by replacing the current component by one of $a, (a+1)$, or $a, 0$, or $0, a$ and incrementing all subsequent non-zero values.

$(=<)$

- Output $\mathbf{m}_a$. Move to a state obtained by replacing the current component by either $0, a+1$ or $a, a+1$ and incrementing all subsequent non-zero a_j. Or, output $\mathbf{r}_a$ and move to the state obtained by replacing the current component by $a, 0$.

$(<=)$

- Output $\mathbf{m}_a$. Move to a state obtained by replacing the current component by either $a, a+1$ or $a, 0$ and incrementing all subsequent non-zero values. Or, output $\mathbf{l}_a$ and move to the state obtained by replacing the current component by $0, a$.

(<<)

- Output $\mathbf{m}_a$. Move to the state obtained by replacing the current component by $a, a+1$ and incrementing all subsequent non-zero values. Or, output $\mathbf{l}_a$ and move to the state obtained by replacing the current component by $0, a$. Or, output $\mathbf{r}_a$ and move to the state obtained by replacing the current component by $a, 0$. Or, output $\mathbf{f}_a$, and move to the state obtained by replacing the current component by $0, 0$ and decrementing all subsequent non-zero values.

That concludes the description of Del except to mention that its initial state is the state 1 and its accepting state is the state corresponding to the empty sequence. Each possible transformation from the insertion encoding of a permutation π to that of some $\sigma \preceq \pi$ is allowed for in the non-deterministic operation of Del and no other transformations are possible. Thus Del does indeed fulfil its specified rôle. $\square$

5 Context free classes

The definition of what it means for a pattern class $\mathcal{C}$ to be represented by a context free language with respect to the insertion encoding is not quite so straightforward as the regular case. Indeed, there are a number of plausible choices for such a definition. We consider only one of them. Already in Av(321) we have seen the necessity of allowing new symbols such as $\mathbf{m}_{-1}$ in the alphabet in order to allow insertions at either end of a configuration. What conditions then shall we demand of a modification to the insertion encoding to define context free pattern classes?

The conditions we will introduce provide a fairly restrictive set of demands. They are centered on the requirements that for some push down automaton $A_{\mathcal{C}}$ which represents $\mathcal{C}$ as a "context free class", each transition of the automaton must correspond to some single letter of the insertion encoding and the number of symbols in the stack is required to equal the number of slots available after the corresponding prefix has been interpreted.

More precisely, let Σ be the alphabet of the automaton $A_{\mathcal{C}}$. Then for each positive integer k (stack size), stack symbol s and state a (of the automaton) we must be given an injective map:

$$T_{(k,s,a)} : \Sigma \to \Lambda_k.$$

These maps must be consistent with respect to stack size. That is, if $T_{(k,s,a)}(x) = \mathbf{f}_i$ for some i then the transition of $A_{\mathcal{C}}$ on symbol x from (k, s, a) should be to $(k-1, s', a')$ and similarly for the other symbols of Λ. Given a word $w = w_1 w_2 \cdots w_n$ accepted by $A_{\mathcal{C}}$ the symbol w_i is processed with some stack size k_i, stack symbol s_i and state a_i. We define the associated word $o_1 o_2 \cdots o_n \in \Lambda^*$ to have $o_i = T_{(k_i,s_i,a_i)}(w_i)$. Then $A_{\mathcal{C}}$ *represents* $\mathcal{C}$ *as a context free class* if the image under this association of its accepted language is the insertion encoding of $\mathcal{C}$.

Definition 15 *Let k be a positive integer. The set $\mathcal{IB}(k)$ consists of all those permutations whose insertion encodings can be written using only symbols whose subscripts come from $\{\pm 1, \pm 2, \ldots, \pm k\}$. We call this the* insertion bounded class of depth k.

Proposition 16 *Each set $\mathcal{IB}(k)$ is a context free pattern class. Its basis consists of the set of permutations of the form:*

$$c_1 a_1 c_2 a_2 \cdots c_k a_k \,(2k+1)\, a_{k+1}, c_{k+1} \cdots a_{2k} c_{2k}$$

where $\{a_1, a_2, \ldots, a_{2k}\} = \{1, 2, \ldots, 2k\}$ and $\{c_1, c_2, \ldots, c_{2k}\} = \{2k+2, 2k+3, \ldots 4k+1\}$.

Proof: The proof of this proposition is essentially the same as that of Proposition 7. That is, consider any permutation π. If the insertion encoding of π includes a symbol of depth more than k, then this symbol, together with representatives of the boundaries of the first and last k slots at the time of its insertion, and elements which fill those slots, provides an involved permutation from the proposed basis. Conversely if one of these permutations is involved in π then the insertion of the symbol corresponding to $2k$ is at depth greater than k. So $\mathcal{IB}(k)$ is a pattern class.

That it is context free follows immediately by simply using a stack whose size is equal to the number of slots available, and that responds to the various symbols in the obvious way. ☐

The basis of $\mathcal{IB}(1)$ is

$$\{41325, 51324, 42315, 52314\}.$$

It is a matter of routine computation to use the description of the push down automaton for $\mathcal{IB}(k)$ to generate its spectrum. For instance, applied to $\mathcal{IB}(1)$ we can determine that the generating function f of this class satisfies:

$$(2t^2 - 2t + 1)f^2 + (4t^2 - 3t)f + 2t^2 = 0$$

and has Taylor series:

$$t + 2t^2 + 6t^3 + 24t^4 + 116t^5 + 632t^6 + 3720t^7 + 23072t^8 + 148528t^9 + \cdots$$

The next result shows that these classes do indeed play the same rôle with respect to context free classes as the classes $\mathcal{SB}(k)$ do for regular classes.

Theorem 17 *Any context free class is a subclass of $\mathcal{IB}(k)$ for some k.*

Proof: Suppose that a class $\mathcal{C}$ with associated push down automaton $A_{\mathcal{C}}$, together with maps $T_{(k,a,s)}$ were a counterexample to this theorem. Then, for any positive integer N there is a permutation in the basis of $\mathcal{IB}(N)$ which belongs to $\mathcal{C}$. At the time that $2N$ is inserted into this permutation the configuration present has $2N+1$ slots. By deleting d elements before the slot in which $2N$ is to be inserted, and $N-d$ after that slot we see that there are configurations c_1 through c_{N+1} each having $N+1$ slots, and such that for each i there is an element of $\mathcal{C}$ whose generation passes through c_i followed by an insertion in the ith slot. Choose $N+1$ to be larger than the product of the sizes of Σ, the stack alphabet and the number of states of the underlying automaton. Then for more than $|\Sigma|$ of the c_i the corresponding triple $(N+1, a_i, s_i)$ is the same. However, different insertion symbols are required for each of these states and only $|\Sigma|$ symbols are available. $\square$

6 Wilfian formulae

Suppose that a family of combinatorial objects with a notion of "size" (a non negative integer) are given, so that there are only finitely many elements in the family of each size. Then a *Wilfian formula* for this family is an algorithm which computes the number of elements in the family of size n in time bounded by a polynomial in n.

Let L be any language in the letters of the insertion encoding that encodes a set of permutations. Although we are generally interested in the case when this set forms a pattern class, this restriction is superfluous at this time. The enumeration problem for L is the same as that for the class of permutations it encodes.

An equivalence relation $\sim_L$ on the set, $\mathrm{Pref}(L)$, of prefixes of words in L can be defined by setting $p \sim q$ if and only if for all $s \in \Lambda^*$,

$$ps \in L \iff qs \in L.$$

In more concrete terms, this equivalence relation can be thought of as relating the configurations that arise in the process of constructing a permutation that lies in the set of permutations encoded by L. Two such configurations are deemed to be equivalent if they have exactly the same successful methods of completing the construction of a permutation in this set. For example, in Section 3 we implicitly made use of this equivalence relation in generating 312-avoiding permutations by noting that the only relevant information about a configuration was the number of slots. This was used to construct an algebraic generating function (and coincidentally a Wilfian formula) for this class. In general we can adapt that methodology in the following way:

Proposition 18 *Suppose that there exists $L' \subseteq \mathit{Pref}(L)$ with the following properties:*

- *L' contains at least one representative from every $\sim_L$ equivalence class.*
- *The number of elements of L' of length n is bounded by a polynomial in n.*
- *There is a polynomial time algorithm which, for each $w \in L'$ and each letter $a \in \Lambda$ for which wa is a prefix of some element of L computes an element $v \in L'$ such that $wa \sim_L v$ and v is not longer than wa.*

Under these conditions L has a Wilfian formula.

Proof: There is little to say about this. It is essentially an application of the transfer matrix approach to enumeration. Suppose that we wish to enumerate the words of length n in L. The conditions allow us to construct in time polynomial in n a directed multigraph G_n with adjacency matrix M_n whose vertices correspond to elements of L' of length at most n and whose edges correspond to insertions of any type provided only that they can be followed by further insertions leading to an element of L. The number of elements of L of length n is then just the number of walks in G_n of length n from the state corresponding to the empty word, to the state corresponding to a configuration with no slots. This can be computed by evaluating the corresponding entry of M_n^n, which is certainly possible in polynomial time. □

A direct implementation of the ideas above shows that if the number of elements of L' of length at most n is $O(n^c)$ then the number of elements of L of length exactly n (in fact the full spectrum of L through length n) can be computed in time $O(n^{c+2})$. Observe also that if L and K are two languages to which the result above applies, then by a similar method (working with the product $L' \times K'$) we also obtain a polynomial time algorithm for enumerating $L \cap K$. Also note that it is possible to relax the third condition somewhat, in particular the length condition on v, though in practice this never seems to be required.

The simplest application of this method is to the class of permutations containing no descending subsequence of length c. In a configuration of say s slots, the only relevant information is the sequence s_i for $0 \leq i < c-1$ where s_i is the number of slots preceding a descending sequence of length i but not one of length $i+1$. Since $s_0 + s_1 + \cdots + s_{c-2} = s$ and $s_0 \leq 1$, there are $O(n^{c-2})$ such sequences. Furthermore, it is easy to check that all the conditions of the proposition above are satisfied. We thus obtain a $O(n^c)$ enumeration algorithm for this class. An algorithm of the same complexity due to Gessel can be found in [13]. However, the latter one requires divisions as well as multiplications and additions/subtractions as well as the evaluation of many terms whose magnitude is significantly larger than the final answer. By contrast the method described above uses only multiplication and addition and (with a little care) can be carried out without ever evaluating a term larger than the final answer.

As mentioned previously in Section 4, the same essential idea was used in [3] applied to the class of 4231 avoiding permutations. In this case there is not a polynomial bound on the number of $\sim_L$ equivalence classes, so an extra limitation restricting the number of slots to some fixed bounds was applied. Although this then enumerates only a proper subclass of Av(4231) this was sufficient to show that $L(4231) > 9$ and thereby refute a conjecture of Arratia.

7 Further applications

7.1 Classes with basis elements of length three and four

A catalogue of the spectra of pattern classes whose basis consists of a permutation of length three and a permutation of length four was given by West in [23]. Most of these were computed using the generating tree approach, with a few exceptions. Although successful, the methods applied in these instances were somewhat *ad hoc*. Using the insertion encoding we can argue that almost all of these classes are regular while the exceptional cases are context free and easily analyzed based on the grammars or stack automata for Av(312) and Av(321).

Using the automorphisms of $(\mathcal{S}, \preceq)$, the length three basis element can always be taken to be either 312 or 321. Consider first a class $\mathcal{C}_\tau = \mathrm{Av}(312, \tau)$ where τ is a permutation of length four not involving 312. Among the basis elements of $\mathcal{SB}(k)$ for $k \geq 2$ only the permutation $(k+1)\,k\,(k+2)\,(k-1)\,\cdots\,2k\,1\,(2k+1)$ avoids 312. Therefore, if for some k this permutation involves τ then the class $\mathcal{C}$ will be regular. This situation applies either to $\mathcal{C}_\tau$ or one of its isomorphs (the isomorphism fixing 312) except for $\mathcal{C}_{1324}$ and $\mathcal{C}_{2143}$.

In both these cases the modifications to the automaton for Av(312) required to enforce the extra restriction are easy to describe. We will deal with the first one only as the second one will arise again in the subclasses of Av(321). Avoiding 1324 implies that after an $\mathbf{l}_1$ or $\mathbf{f}_1$ (creating a "1") no further $\mathbf{m}_1$'s may occur (that would be the 2, and the restriction that all insertions be in the first slot would lead inevitably to a subsequent 3 and 4), nor may an $\mathbf{r}_1$ occur until only a single slot remains. These conditions transform naturally into a grammar for this class:

$$\begin{array}{rcl} S & \rightarrow & \mathbf{f} \,|\, \mathbf{l}T \,|\, \mathbf{r}S \,|\, \mathbf{m}UT \\ U & \rightarrow & \mathbf{f} \,|\, \mathbf{l}V \,|\, \mathbf{r}U \,|\, \mathbf{m}UV \\ V & \rightarrow & \mathbf{f} \,|\, \mathbf{l}V \\ T & \rightarrow & \mathbf{f} \,|\, \mathbf{l}T \,|\, \mathbf{r}T. \end{array}$$

In this grammar, S as usual represents the language, U represents an unrestricted first slot, V a slot other than the rightmost slot which has been restricted by the placement of a small element to the left, and T the rightmost slot after such a restriction. Solving the resulting system of equations for the generating function of the class yields:

$$\frac{t - 3t^2 + 3t^3}{(1-t)(1-2t)^2}.$$

Now consider classes $\mathcal{D}_\tau = \mathrm{Av}(321, \tau)$ where τ is of length 4. For $k \geq 3$ the only elements of the basis of $\mathcal{SB}(k)$ that avoid 321 are those of the form $bab \cdots bab$ where the a's form an increasing sequence, and so do the b's except possibly for the final element. This situation applies to $\mathcal{D}_\tau$ or one of its isomorphs except for the case $\tau = 2143$. Recall that in the modified insertion encoding of Av(321) we were restricted to splitting or adding on the left of the final slot, or filling or adding on the left of the first slot. Symbols of the latter type create a 21 (possibly spanning the first slot) and require the remaining slots to be filled strictly from left to right (to avoid 2143) except in *precisely* the case of a configuration of the form $2\diamond 1\diamond$ (where 1 might stand for an increasing sequence of values smaller than 2). In this case, we may add on the left hand end of either slot. This condition complicates the grammar somewhat and gives:

$$
\begin{array}{rcl}
S & \rightarrow & \epsilon \,|\, \mathbf{l}_{-1}S \,|\, \mathbf{m}_{-1}AT \\
A & \rightarrow & \mathbf{l}_{-1}A \,|\, \mathbf{m}_{-1}BY \,|\, \mathbf{f}_1 \,|\, \mathbf{l}_1X \\
B & \rightarrow & \mathbf{l}_{-1}B \,|\, \mathbf{m}_{-1}BY\; F_1 \,|\, \mathbf{l}_1Y \\
X & \rightarrow & \mathbf{f}_1 \,|\, \mathbf{l}_1X \,|\, \mathbf{l}_{-1}X \\
Y & \rightarrow & \mathbf{f}_1 \,|\, \mathbf{l}_1Y \\
T & \rightarrow & \epsilon \,|\, \mathbf{l}_{-1}T
\end{array}
$$

In this grammar, S as usual represents the language, A represents the first newly created slot, X, B any subsequently created slot, X the fashion in which the first newly created slot can be filled if the configuration $2\diamond 1\diamond$ is created by the first occurrence of $\mathbf{l}_1$, Y the fashion in which any new slots can be filled if this is not the case, and T the sequence of elements added when we reach a configuration equivalent to $21\diamond$. This yields a generating function for this class of:

$$\frac{1-5t+10t^2-9t^3+4t^4}{(1-2t)(1-t)^4}$$

7.2 Generalised patterns

A number of generalisations of pattern avoidance have appeared in the literature [9, 10, 21]. Two of the more common are *blocked patterns* (also called gapped patterns) and *barred patterns*. In an occurrence of a blocked pattern, certain parts of the pattern must occur as consecutive elements. The blocks are generally separated by -'s or *gaps* so for instance an occurrence of $1-23$ is an occurrence of 123 in the ordinary sense where the second and third elements are consecutive. In a barred pattern certain patterns are forbidden *unless* they occur within other patterns. Thus an occurrence of $31\bar{4}2$ is an occurrence of 312 with no element between the symbols representing the 1 and the 2 that is larger than the symbol representing 3. That the two notions are not entirely complementary can be seen by noting that avoiding $31\bar{4}2$ is in fact equivalent to avoiding $3-12$. For in any occurrence of 312 which is not a $31\bar{4}2$ the elements between the 1 and 2 (inclusive) must all be strictly smaller than the 3. Since this sequence of elements finishes with a larger element than it starts with it must include at least one ascent. This ascent gives an occurrence of $3-12$. Conversely, any occurrence of $3-12$ is already an occurrence of $31\bar{4}2$.

Claesson in [9] showed that the classes avoiding a single gapped pattern of length 3 divide into two Wilf-equivalent groups, one enumerated by the Bell numbers and the other by the

Catalan numbers. In the first group, natural symmetries (reversal and complementation) account for all of the equivalences except those between the classes $3-12$ and $3-21$.

Using the insertion encoding, this Wilf equivalence can be directly illustrated. In each case, knowing that the answer is the Bell numbers, it is relatively easy to construct a bijection with set partitions thus "explaining" the Wilf equivalence. However, the bijections provided by the insertion encoding can be seen *without* knowing the actual enumeration sequence.

Consider a configuration that arises in the construction of a $3-12$ avoiding permutation. We work in the modified insertion encoding where the right hand slot must remain unfilled (i.e. only $\mathbf{l}$ and $\mathbf{m}$ operations are permitted in the rightmost slot). The next element added can be part of a 12 only if it is added using $\mathbf{f}$ or $\mathbf{l}$. Such an addition will inevitably be part of a $3-12$ pattern, unless it occurs in the first slot. So, if there are $n > 1$ slots available, the next code symbol in a $3-12$ avoider can be any one of $\mathbf{f}_1$, $\mathbf{l}_1$, $\mathbf{r}_i$ for $1 \leq i < n$ and $\mathbf{m}_i$ for $1 \leq i \leq n$. Counting transitions gives 1 transition to $n-1$ slots, n transitions to n slots, and n transitions to $n+1$ slots. These numbers (except for the first) also apply to the configuration consisting of a single slot.

Now consider the analysis of $3-21$ avoiding permutations in the same way. This time any $\mathbf{r}$ at all or any $\mathbf{f}$ except in the first slot will inevitably create an occurrence of $3-21$. So, if there are $n > 1$ slots available, the next code symbol in a $3-21$ avoider can be any one of $\mathbf{f}_1$, $\mathbf{l}_i$ for $1 \leq i \leq n$ and $\mathbf{m}_i$ for $1 \leq i \leq n$. Again, the pattern is consistent with the one slot case. Since the number of transitions from each slot configuration agree in the two cases, the two classes are Wilf equivalent. Moreover, we have an explicit bijection between the classes which amounts to changing any $\mathbf{r}$ occurring in the generation of a $3-12$ avoiding permutation into an $\mathbf{l}$ (with the same subscript).

7.3 Classes closed in the strong Bruhat order

The strong Bruhat order is a partial order on permutations defined by taking the transitive closure of relations of the form:

$$\alpha a \gamma b \delta < \alpha b \gamma a \delta$$

whenever $b < a$. Investigation of pattern classes closed downwards in the strong Bruhat order was begun in [1] motivated by the consideration of abstract machines whose purpose it is to sort permutations.

Definition 19 *Let r and s be positive integers. The permutation class $\mathcal{B}(r,s)$ consists of all those permutations not containing a subsequence of the form:*

$$b_1 b_2 \cdots b_r a_1 a_2 \cdots a_s$$

where $b_i > a_j$ for all $1 \leq i \leq r$ and $1 \leq j \leq s$. Equivalently, it is is the class whose basis consists of the $r!s!$ permutations of length $r+s$ which are precisely of this form.

In [1] the following result appears:

Proposition 20 *Let $\mathcal{C}$ be a non trivial pattern class closed downwards in the strong Bruhat order. Then $C \subseteq \mathcal{B}(r, s)$ for some r and s.*

We therefore obtain:

Corollary 21 *Let $\mathcal{C}$ be a non trivial pattern class closed downwards in the strong Bruhat order. Then $C \subseteq \mathcal{SB}(k)$ for some k.*

Proof: Recall that $\mathcal{SB}(k)$ consists of all those permutation whose evolution through the insertion encoding never requires more than k slots at a time. Its basis consists of the $k!(k+1)!$ permutations of the form:

$$baba \cdots ab$$

where the b's form the set $\{k+1, k+2, \ldots, 2k+1\}$ and the a's form the set $\{1, 2, \ldots, k\}$. From the form of the basis elements it follows that for all positive integers r and s, $\mathcal{B}(r, s) \subseteq \mathcal{SB}(2r+2s-1)$ since every basis element of the latter class involves at least one basis element of the former class. Taking r and s as provided by the previous proposition completes the proof. $\square$

Recall that Theorem 8 states that any pattern class contained in $\mathcal{SB}(k)$ for some k whose basis is encoded by a regular language in the insertion encoding is itself regular. Such classes automatically have rational generating functions. For a class closed downwards in the strong Bruhat order there are potentially two definitions of "basis". We might intend a minimal set of forbidden patterns in the ordinary sense, or we might intend a minimal set of forbidden patterns with respect to the transitive closure of the involvement and strong Bruhat orders. Call the former type the *ordinary basis* and the latter type the *strong basis*. In [1] it is shown (Theorem 14) that if the strong basis is finite then so is the ordinary basis (the converse is trivial). So we obtain:

Proposition 22 *If a class $\mathcal{C}$ closed downwards in the strong Bruhat order has a finite strong basis then it has a rational generating function.*

In [1] the generating functions for all such classes having a single strong basis element of length at most four are determined. Although the preceding result is in principle effective, it proved simpler to deal with these situations on a case by case basis using the structural information about the elements of these classes determined by their ordinary bases.

7.4 Miscellaneous examples

As soon as we consider classes with a single basis element of length four, or with a pair of basis elements of length four it is a rare occurrence that these are context free in the restricted sense defined above. An exception are a number of the pairs of permutations of length four which are bases for classes whose enumeration sequence consists of the large Schröder numbers (sequence A006318 of [17]). Building on earlier work of West [22], Kremer showed in [15] that ten fundamentally different pairs of permutations of length four all generate classes enumerated by the large Schröder numbers.

Many of these classes are easily seen to be equinumerous by recourse to their insertion encoding. We will illustrate this with a single example, which we grant is the most attractive and simple. Consider the two classes:

$$\begin{aligned} \mathcal{C}_1 &= \mathrm{Av}(3124, 4123) \\ \mathcal{C}_2 &= \mathrm{Av}(3142, 4132). \end{aligned}$$

Then arguments very similar to those of Section 3 and to the proofs of Propositions 7 and 16 show that with respect to the insertion encoding these two classes are defined by the following rules:

$\mathcal{C}_1$ Any operation is allowed in the first slot. Additionally if there are two or more slots $\mathbf{f}_{-1}$ and $\mathbf{r}_{-1}$ are allowed.

$\mathcal{C}_2$ Any operation is allowed in the first slot. Additionally if there are two or more slots $\mathbf{f}_2$ and $\mathbf{l}_2$ are allowed.

We will give details of the argument for the first class only. First of all, the two proposed basis elements violate the given conditions as in both cases the 2 is added with $\mathbf{l}_2$. So $\mathcal{C}_1 \subseteq \mathrm{Av}(3124, 4123)$. For the reverse containment we show as usual that any permutation whose insertion encoding violates one of the given conditions must involve one of these two elements. The two conditions are equivalent to the single condition: no insertion is allowed which leaves a slot to its right, except in the first slot. Suppose that such an insertion is made, say of an element b. At that point we have the situation $\diamond\, a \cdots b \cdots \diamond$ within the current configuration, where a is any element forming part of the right hand boundary of the first slot. Regardless of how those two slots are filled we will obtain either *cabd* or *dabc* i.e. one of the two forbidden patterns.

From the description of the two classes it follows immediately that they are equinumerous. We can mechanically derive the (known) generating function by the following method (based on techniques described in [12]) which demonstrates a skeleton for such arguments that apply to any class where the allowed insertions are always the same once a certain number of slots are present.

Suppose that there are $k \geq 2$ slots. Then two of the allowed symbols reduce the number of slots by one, three leave the number of slots unchanged, and one increases the number of slots by one. If we let $f_{\geq 2}(t)$ be the generating function for sequences of symbols which start from $k \geq 2$ slots, never pass through a configuration with fewer than k slots, and return for the first time to a configuration of k slots at the end, then:

$$f_{\geq 2} = 3t + t\left(\frac{1}{1 - f_{\geq 2}}\right) 2t.$$

The derivation of this equation is by the observation that such sequences include:

- three sequences of length one which do not change the number of slots, and otherwise,
- an initial increase by one slot, followed by a sequence (possibly empty) of returns to this number of slots without decreasing the number of slots, followed by a decrease of one slot.

From the initial configuration containing a single slot there are only two transitions which leave the number of slots unchanged. So the equation for the generating function g counting first return to the one slot configuration is:

$$f_1 = 2t + t\left(\frac{1}{1 - f_{\geq 2}}\right) 2t.$$

Finally, the encodings of elements of the class consist of a sequence of returns to the one slot configuration, followed by the final fill, i.e. $f = t/(1 - f_1)$. Solving this set of equations gives (of course) the large Schröder numbers.

However, we can also make use of the insertion encoding to enumerate $\mathcal{C}_1 \cap \mathcal{C}_2 = \text{Av}(3124, 4123, 3142, 4132)$. For this class is described by the rules: any operation in the first slot is permitted, as is filling the second slot if there are precisely two slots. Using the same approach to finding the generating function (starting from configurations with $k \geq 3$ slots) shows that:

$$|\mathcal{C}_1 \cap \mathcal{C}_2 \cap \mathcal{S}_n| = \binom{2n-2}{n-1}.$$

A result of applying this same method was reported in [1]. The generating function,

$$\frac{3 - 13t + 2t^2 + (5t-1)\sqrt{1-4t}}{2(1 - 4t - t^2)},$$

of the class $\text{Av}(3241, 3421, 4321)$ was computed by representing it as a context free language with respect to the insertion encoding. This class is of interest because it is the largest permutation class not containing the permutation 3241 that is closed under the operation: $\alpha ba\beta \to \alpha ab\beta$ for $b > a$ which exchanges any adjacent pair of elements in a permutation that are not in their correct value order. That is, it is a class in the *weak Bruhat order* whose weak basis is 3241.

8 Conclusions

The insertion encoding provides a new tool for the analysis of pattern closed classes of permutations. Our presentation has concentrated on its underlying theory, and a few specific examples connected to enumeration and Wilf equivalence to illustrate its use as a unifying framework for known enumeration results in this field. On the other hand it has already shown its utility in the refutation of Arratia's conjecture [3], and in the enumeration of a number of classes closed downwards in the weak Bruhat order as well as providing general results about finitely based classes closed downwards in the strong Bruhat order [1].

As we have seen, the insertion encoding can often be used to provide a polynomial time enumeration algorithm for pattern classes. The Noonan-Zeilberger (or Gessel-Noonan-Zeilberger) conjecture asserts that any finitely based pattern class will have a P-recursive generating function, so the concentration on such formulae may seem a little misplaced. However, the recently produced data on the number of 4231-avoiding permutations has caused some (including Zeilberger himself [25]) to cast doubt on this conjecture. In that context such algorithms regain a measure of importance.

The general theory of pattern classes whose insertion encodings form a regular language seems to be well understood. For most "interesting" classes in this group, which necessarily have relatively short basis elements and are restricted to making use of a small number of slots, the effective methods provided are sufficient to perform any computations desired in these classes. However, it would definitely be of interest to determine whether more efficient algorithms for computing in these classes are available, or alternatively to establish that some problems in this family are NP-hard.

The situation with respect to context free languages is not nearly so clear. In particular, the astute reader will have noticed that results corresponding to Theorem 8, Corollary 10 or Proposition 13 are not presented in this paper. It seems to us that a result of the latter type should be relatively easy to prove. There seems to be some hope of proving at least certain restricted cases of Corollary 10 in the context free case. However, results corresponding to Theorem 8 seem unlikely, given the weak closure properties of context free languages. Moreover, the definition of context-free class that we provided above is only one possible one. It does not encompass such classes as the *separable* permutations (those avoiding both 2413 and 3142) whose natural recursive structure cries out for such an identification. With regard to this specific class we can say that by paying suitable attention to the tree structure that exists among the slots of a configuration (where slots created by an $\mathbf{m}$ are considered to be children of the slot in which the $\mathbf{m}$ took place) it is possible to provide a context free interpretation for the insertion encodings of elements of this class. However, this interpretation is quite convoluted in comparison to the natural recursive description of the class. Of course this may simply be taken as an indication of the rather obvious, and heartening, observation that no single technique of describing permutations is ever likely to handle all pattern classes with equal facility.

The application of the insertion encoding to the description of generalized pattern classes has only been touched upon here, and is the subject of further investigation. Generalizations of the insertion encoding can be used to enumerate other families of combinatorial objects, for example matchings on $\{1, 2, \ldots, 2n\}$ avoiding certain patterns.

We believe the insertion encoding to be a significant new tool in the investigation of the structural and enumerative properties of pattern classes and their relatives. As well as providing some general theory, it unifies the presentation of a number of known results, and is useful in the solution of specific exact and approximate enumerative problems.

References

[1] M. H. Albert, R. E. L. Aldred, M. D. Atkinson, H. P. van Ditmarsch, C. C. Handley, D. A. Holton, and D.J. McCaughan. Sorting classes. Tech Report OUCS-2005-04, http://www.cs.otago.ac.nz/research/techreports.html, 2005.

[2] M. H. Albert, M. D. Atkinson, and N. Ruškuc. Regular closed sets of permutations. *Theoret. Comput. Sci.*, 306(1-3):85–100, 2003.

[3] M. H. Albert, M. Elder, A. Rechnitzer, P. Westcott, and M. Zabrocki. On the Wilf-Stanley limit of 4231-avoiding permutations and a conjecture of Arratia. http://www.arxiv.org/abs/math.CO/0502504, 2005.

[4] Richard Arratia. On the Stanley-Wilf conjecture for the number of permutations avoiding a given pattern. *Electron. J. Combin.*, 6:Note, N1, 4 pp. (electronic), 1999.

[5] Silvia Bacchelli, Elena Barcucci, Elisabetta Grazzini, and Elisa Pergola. Exhaustive generation of combinatorial objects by ECO. *Acta Inform.*, 40(8):585–602, 2004.

[6] Cyril Banderier, Mireille Bousquet-Mélou, Alain Denise, Philippe Flajolet, Danièle Gardy, and Dominique Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Math.*, 246(1-3):29–55, 2002. Formal power series and algebraic combinatorics (Barcelona, 1999).

[7] Miklós Bóna. *Combinatorics of permutations*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2004.

[8] N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In *Computer programming and formal systems*, pages 118–161. North-Holland, Amsterdam, 1963.

[9] Anders Claesson. Generalized pattern avoidance. *European J. Combin.*, 22(7):961–971, 2001.

[10] Anders Claesson and Toufik Mansour. Counting occurrences of a pattern of type $(1,2)$ or $(2,1)$ in permutations. *Adv. in Appl. Math.*, 29(2):293–310, 2002.

[11] Enrica Duchi, Jean-Marc Fedou, and Simone Rinaldi. From object grammars to ECO systems. *Theoret. Comput. Sci.*, 314(1-2):57–95, 2004.

[12] P. Flajolet. Combinatorial aspects of continued fractions. *Discrete Math.*, 32(2):125–161, 1980.

[13] Ira M. Gessel. Symmetric functions and P-recursiveness. *J. Combin. Theory Ser. A*, 53(2):257–285, 1990.

[14] Donald 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.

[15] Darla Kremer. Permutations with forbidden subsequences and a generalized Schröder number. *Discrete Math.*, 218(1-3):121–130, 2000.

[16] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Combin. Theory Ser. A*, 107(1):153–160, 2004.

[17] N. J. A. Sloane. The online encyclopedia of integer sequences, 2005. `http://www.research.att.com/~njas/sequences/`.

[18] Vincent Vatter. Finitely-labeled generating trees and restricted permutations. To appear, J. Symb. Comp. `http://www.arxiv.org/abs/math.CO/0309238`.

[19] Vincent Vatter. Enumeration schemes for restricted permutations. `http://www.math.rutgers.edu/~vatter/publications/wilfplus/`, 2005.

[20] Vincent Vatter. WilfPlus. `http://www.math.rutgers.edu/~vatter/programs/wilfplus/`, 2005.

[21] Julian West. Sorting twice through a stack. *Theoret. Comput. Sci.*, 117(1-2):303–313, 1993. Conference on Formal Power Series and Algebraic Combinatorics (Bordeaux, 1991).

[22] Julian West. Generating trees and the Catalan and Schröder numbers. *Discrete Math.*, 146(1-3):247–262, 1995.

[23] Julian West. Generating trees and forbidden subsequences. *Discrete Math.*, 157(1-3):363–374, 1996.

[24] Doron Zeilberger. Enumeration schemes and, more importantly, their automatic generation. *Ann. Comb.*, 2(2):185–195, 1998.

[25] Doron Zeilberger. Personal communication. Comments made at the *Permutation Patterns* conference, 2005.