Compositions of pattern restricted sets of permutations

M. H. Albert[*] R. E. L. Aldred[†] M. D. Atkinson[*]
H. P. van Ditmarsch[*] C. C. Handley[*] D. A. Holton[†]
D. J. McCaughan[†]

September 3, 2004

Abstract

The composition of two pattern restricted classes X, Y is the set of all permutation products $\theta\phi$ where $\theta \in X, \phi \in Y$. This set is also defined by pattern restrictions. Examples are given where this set of restrictions is finite and where it is infinite. The composition operation is studied in terms of machines that sort and generate permutations. The theory is then applied to a multistage sorting network where each stage can exchange any number of adjacent disjoint pairs.

1 Introduction

In the theory of pattern restricted permutations it is normal to regard permutations as lists of integers whereas, in algebra, permutations are regarded as mappings. In this paper we shall explore pattern restricted sets of permutations from an algebraic viewpoint. The notion that allows us to link the combinatorial and algebraic aspects is that of a *permuting machine*. Such a machine accepts a stream of tokens and generates an output stream that is a permutation of the input subject to the following two conditions:

1. The names (values) of the input items are unimportant. For example, the possible behaviours of the machine on the input $3, 1, 4, 2$ are the same as on the input $1, 2, 3, 4$ (e.g. if it can reverse one input it can reverse the other).

[*]Department of Computer Science, University of Otago
[†]Department of Mathematics and Statistics, University of Otago

2. Suppose that the machine is able to transform an input α into an output β and that α' is a subsequence of α that becomes rearranged as the subsequence β' of β. Then the machine must be able to transform α' (if presented as an input sequence in its own right) into β'.

Because of the first property we can, if we wish, take the input stream to be $1, 2, \ldots, n$ and treat the machine as a device for generating permutations; or we can take the output stream to be $1, 2, \ldots, n$ and treat it as a device for sorting permutations. In this paper we shall normally adopt the former stance.

The second property shows that permuting machines are closely related to the theory of permutation patterns for it implies that if a permutation σ can be generated by a machine then so can any permutation whose "pattern" occurs in σ as a subsequence. The notion of permutation pattern containment (also called *involvement*) is formally defined as follows: π is involved in σ (written $\pi \preceq \sigma$) if σ has a subsequence σ' whose terms are ordered in the same relative way as those of π. For example, $3142 \preceq 152463$ because the latter permutation has the subsequence 5263. Then, directly from the definitions, we have

Proposition 1 *If X is the set of permutations that can be generated by the permuting machine M then X is closed under involvement in the sense that $\sigma \in X$ and $\pi \preceq \sigma$ implies $\pi \in X$.*

The first significant permuting machine to be studied was the stack in Knuth's *Art of Computer Programming* series [7]. The associated *stack permutations* are those output sequences that can be obtained by submitting the input $1, 2, \ldots, n$ to a stack and performing a sequence of push and pop operations. Knuth proved two results that more than three decades later still inspire the now active combinatorial area of pattern involvement. His first result was that a permutation is a stack permutation if and only if it does not involve the permutation 312.

For a general closed set X we can consider set $B(X)$ of permutations not lying in X and minimal with respect to involvement. This is called the *basis* of X and one easily verifies that a permutation lies in X if and only if it avoids (does not involve) every member of the basis. In this terminology Knuth had proved that $\{312\}$ is the basis of the set of stack permutations. His second result was a determination of the number of stack permutations of each length. Similar enumeration problems for other closed sets are still being studied.

Other examples of permuting machines which have been studied from the same standpoint include deques [9], networks of stacks and queues [12], stacks in series [6], and packet switching networks [5]. For each of these the same two questions are attacked: what is the basis of the associated closed set, and how many permutations of each length does it contain?

Permuting machines add an algebraic dimension to closed sets when we combine them in series channelling the output of one machine to the input of another.

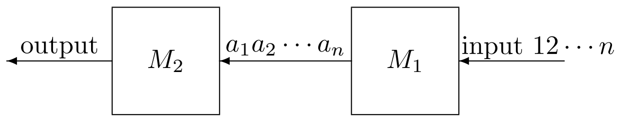

Figure 1: 2 machines in series

This serial combination is also a permuting machine as depicted in Figure 1. The following result is clear.

Proposition 2 *Suppose that machines M_1, M_2 are associated with sets X_1, X_2 of permutations. Then the serial combination of them is associated with the set $X_2 \circ X_1$ of permutations that are compositions of permutations in X_2 with permutations in X_1.*

We will call $X \circ Y$ the composition of X and Y. It is a closed set but working out its properties from those of X and Y is not easy and only a handful of previous results have been found [3]. The difficulties stem from $X \circ Y$ being much more complex than either of X and Y. An example of this is the composition $S^2 = S \circ S$ where S is the set of stack permutations. Although S itself has been very well understood since the 1960s, virtually nothing was known about the set of permutations generated by two stacks in series until Murphy [8] proved that it was not finitely based. The enumeration question for S^2 is still open.

In Section 2 we shall collect some more positive results by identifying some compositions that can be computed. In Section 3 we shall study a composition that is associated with a sorting network that operates in a fixed number of rounds; in each round a set of transpositions are effected. Finally, we record a miscellany of smaller results and pose some open problems.

2 Some tractable compositions

In this section we shall study some families of closed sets that have figured in previous studies and prove enough about their various compositions that enumeration information and basis information about them can be readily obtained.

2.1 Profile sets

Let σ be some fixed permutation of length k. The profile set $P(\sigma)$ associated with σ consists of all those permutations that can be obtained from σ by replacing each of its terms of σ by an increasing segment of consecutive terms (which may be empty) so that the relative ordering of the segments is the same as the terms of P. For example $567123489 \in P(213)$. Profile sets were introduced in

[2] as a tool for classifying closed sets. They are known to be finitely based and their enumeration problems have been solved.

When discussing a profile set $P(\sigma)$ we may consider σ to be *irreducible* in the sense that it contains no segment $i, i+1$ (if that happened we could replace σ by a shorter permutation). A profile set obviously has the property that it is invariant under the "expansion" operations that replace any symbol t of one of its permutations by the segment $t, t+1$ (with appropriate renumbering of the symbols greater than t). For brevity a set with this property is said to be *invariant under expansion*. Conversely, any set which is invariant under expansion is a union of profile classes; the number of profile classes in this this union is finite if the number of irreducible permutations is finite.

A profile set is naturally associated with a profile machine $PM(\sigma)$. The machine $PM(\sigma)$ divides its input into k segments (some of which may be empty) and permutes the segments bodily according to the permutation σ. The following theorem determines the permutations generated by the serial combination of two profile machines.

Theorem 3 *The composition $P(\sigma) \circ P(\tau)$ is a finite union of profile sets.*

To prove the theorem we need the following two lemmas.

Lemma 4 *$P(\sigma) \circ P(\tau)$ contains only a finite number of irreducible permutations.*

Proof: The permutations of $P(\sigma) \circ P(\tau)$ arise from combining the machines $PM(\tau)$ and $PM(\sigma)$ in series and applying the composite machine to the input $1, 2, \ldots, n$. To begin with $PM(\tau)$ outputs a sequence with $|\tau|$ segments (some possibly empty) each being an ascending sequence of consecutive values. Then $PM(\sigma)$ causes this output to be divided into $|\sigma|$ segments and permutes them. The result consists of at most $|\sigma||\tau|$ segments (the intersections of the segments defined by $PM(\sigma)$ with those defined by $PM(\tau)$) each an ascending sequence of consecutive values. But if this is irreducible such segments must have length at most 1. ■

A slightly more careful argument of the segment intersection in this proof allows us to replace the bound $|\sigma||\tau|$ by $|\sigma| + |\tau| - 1$.

Lemma 5 *Let X, Y be closed sets each invariant under expansion. Then also $X \circ Y$ is invariant under expansion.*

Proof: In this proof it is more convenient to use notations associated with regarding permutations as mappings. In particular we shall write i^α for the image of i under the permutation α. We begin the proof by expressing the expansion of a permutation in an algebraic form. Let $\alpha = a_1 a_2 \ldots a_{n-1}$ be a permutation of length $n-1$ and let $\hat{\alpha}$ be the permutation of length n obtained from α by replacing a_i by $a_i\ a_i+1$ (and incrementing any term greater than a_i). Then a direct calculation verifies that

$$\hat{\alpha} = C_i^{-1} \circ \alpha n \circ C_{i^\alpha}$$

where, in general, C_k denotes the cycle $(k+1, k+2, \ldots, n)$.

Now let $\alpha \in X$ and $\beta \in Y$ be permutations of length $n-1$. We wish to show that all expansions of $\alpha \circ \beta$ lie in $X \circ Y$. A typical such expansion has the form $C_i^{-1} \circ (\alpha \circ \beta)n \circ C_{i^{\alpha\circ\beta}}$. But

$$\begin{aligned} C_i^{-1} \circ \alpha\beta n \circ C_{i^{\alpha\circ\beta}} &= C_i^{-1} \circ \alpha n \circ \beta n \circ C_{i^{\alpha\circ\beta}} \\ &= (C_i^{-1} \circ \alpha n \circ C_{i^\alpha}) \circ (C_{i^\alpha}^{-1} \circ \beta n \circ C_{i^{\alpha\circ\beta}}) \end{aligned}$$

which is the product of an expanded permutation in X with an expanded permutation in Y. Since X and Y are themselves invariant under expansions the product lies in $X \circ Y$ as required. ■

The proof of Theorem 3 follows directly from these two lemmas. Notice also that the method of proof is constructive and shows how the profile subsets of $P(\sigma) \circ P(\tau)$ could be found. It also follows, from the results of [2], that $P(\sigma) \circ P(\tau)$ is finitely based and has a polynomial enumeration function.

2.2 W sets

A *signature* is defined to be a sequence $\epsilon = (e_1, e_2, \ldots, e_k)$ of signs $+1, -1$. For every signature we define a set $W(\epsilon)$ of all those permutations which are the concatenations of k segments $\sigma_1, \sigma_2, \ldots, \sigma_k$ (some of which may be empty) where σ_i is increasing if $e_i = +1$ and decreasing if $e_i = -1$. For example, $3589\ 421\ 67 \in W(1, -1, 1)$. Such sets were studied in [4] as examples of partially well-ordered closed sets and again in [1] as examples of sets with encodings as a regular language. The set $W(\epsilon)$ is associated with a machine $WM(\epsilon)$ that first divides its input stream into k subsequences; those that correspond to -1 in the signature are reversed; then all the subsequences are concatenated.

Theorem 6 *Let δ, ϵ be two signatures and let $\delta = d_1 d_2 \ldots$. Then $W(\delta) \circ W(\epsilon) = W(\eta)$ where the signature η is defined by*

$$\eta = \epsilon^{d_1} \epsilon^{d_2} \cdots$$

and where $\epsilon^1 = \epsilon$ and ϵ^{-1} is obtained by reversing and negating ϵ.

Proof: The permutations of $W(\delta) \circ W(\epsilon)$ are obtained by the serial combination of a machine $WM(\epsilon)$ with a machine $WM(\delta)$. The $WM(\delta)$ machine takes a sequence structured as an ϵ signature, divides it into subsequences each of which will also be structured with an ϵ signature, reverses those corresponding to the terms $d_i = -1$ and concatenates the results. This gives a result with the signature stipulated in the theorem.

To prove the converse we prove that every permutation of the stated signature can be undone by an appropriate pair of undoing machines in series. Undoing an operation of some signature $\tau = t_1 t_2 \cdots t_r$ is performed by splitting the input into r segments, reversing those that correspond to $t_i = -1$ and merging the segments. So what we have to do here is split the permutation according to the segments ϵ^{d_i}, reverse those we need to, and do a merge so that each of the segments (now, after the reverses, all of signature ϵ) are merged so that the individual segments of each ϵ structured segment get merged into order (increasing or decreasing as appropriate). The result has signature ϵ. ■

Corollary 7 *Let X and Y be closed subsets of $W(\delta)$ and $W(\epsilon)$. Then $X \circ Y$ is finitely based and has a rational generating function.*

Proof: $X \circ Y$ is a subset of some $W(\eta)$ and the result now follows from Theorem 4 of [1]. ■

2.3 Regular sets

In this section we study compositions using ideas from the theory of finite automata. Given a permutation $\sigma = s_1 \dots s_n$ we define the *rank* of a term s_i to be its relative value in the set $\{s_i, s_{i+1}, \dots, s_n\}$ (so, if the rank is ℓ say, there will be $\ell - 1$ terms in $\{s_{i+1}, \dots, s_n\}$ that are smaller than s_i). A permutation is said to be *k-bounded* if every term has rank at most k. For example 243615 is 3-bounded.

Every k-bounded permutation $\sigma = s_1 \dots s_n$ can be encoded as a word $e_1 \dots e_n$ in the alphabet $\{1, \dots, k\}$ where e_i is the rank of s_i. For example, 243615 encodes as 232311.

In [1] a closed set of k-bounded permutations was defined to be *regular* if its encoding was a regular set in the sense of finite automata. For our purposes it is convenient to use non-deterministic finite automata whose associated language is the set of words that they can output rather than the set of words that they accept as inputs. .Regular closed sets of permutations have many special properties [1]: they have linear time recognisers, and algorithms based on finite automata for computing their bases and their generating functions (which are always rational).

In this subsection we shall prove

Theorem 8 *The composition of two regular sets of permutations is regular.*

Before giving the proof of the theorem we shall describe our principal tool: a new type of machine called a *finite state permuter*. Such a machine has a finite set M of *modes* and, at each point of its operation, is in one of these modes. It also has a finite set of k registers $R_1, R_2, \ldots, R_k$ each of which is either empty or holds one symbol from the input; in general the symbols held in the registers are the next k_0 symbols ($k_0 \leq k$) of the input (that have not yet been output). The *disposition* of the machine is the content of $R_1, \ldots, R_k$ up to relative order of occurrence within the input stream; therefore the set D of dispositions is finite. One of the modes m_0 is the initial mode and the disposition d_0 where all the registers are empty is the initial disposition. The state set of the machine is $M \times D$ (which is finite) and the initial state is (m_0, d_0).

The machine begins in its initial state, undergoes a number of transitions, and halts when all the input has been transferred to the output. Each transition is controlled by a non-deterministic transition function τ that, given the current state, determines the next state and possibly determines one of the registers which then discharges an output symbol. The new disposition of the machine may require that another symbol in the input stream be transferred to one of the registers.

Every computation of the permuting machine results in an output that is a rearrangement of the input. Since the names of the input symbols do not figure in the description (only their positions within the input stream) the machine has the first property demanded of a permuting machine. Hence we may, when convenient, take the input stream to be $1, 2, \ldots, n$ and consider the possible output streams to be the set of permutations computed by the machine. There is no guarantee that the second property holds so this set of permutations may not be closed.

Lemma 9 *The set X of permutations that can be output by a finite state permuter P has a regular encoding.*

Proof: X is the set of permutations that P can output if presented with input $1, 2, \ldots, n$. From the finite state permuter P we can construct an ordinary finite state machine F that outputs the encoded form of X. The set of states will be the same in each. The only difference between P and F is that F outputs, rather than a symbol s, the rank of s within the set of symbols that have not yet been output. However this rank is determined by the transition function of P (which picks a register) and the current disposition of symbols within the registers. Thus an appropriate transition function for F can be defined. However, F is a finite state machine whose output language is the encoded form of X and this completes the proof. ∎

We now prove a converse result.

Lemma 10 *Let X be a set of permutations whose encoding is regular. Then there is a finite state permuter that defines it.*

Proof: Let $E(X)$ be the set of words in the symbols $\{1, \ldots, k\}$ that are the encodings of the permutations of X. Since $E(X)$ is a regular set there is a non-deterministic finite automaton A that outputs the words of $E(X)$. Suppose that Q is the set of states, q_0 is the initial state of A, q_f is the final state, and $\tau : Q \to Q \times \{1, \dots, k\} \cup \epsilon$ the transition function.

We define a finite state permuter P with k internal registers and specify how it operates when presented with an input stream $1, 2, \dots, n$. The modes of P are the states of A and the dispositions are simply that, for some $k_0 \leq k$, $R_1, \dots, R_{k_0}$ hold the next k_0 symbols that have not yet been output, and the remaining registers are unoccupied.

Initially P has transitions that insert the first k symbols into the registers in order. Subsequently, if any register R_j discharges an output symbol then the contents of $R_{j+1}, \dots, R_k$ are moved to $R_j, \dots, R_{k-1}$ and the next input symbol is placed in R_k.

The subsequent transitions of P mirror those of A. In other words, whenever A outputs a symbol j with $1 \leq j \leq k$, P outputs the symbol in register R_j. The machine P changes mode according to how A changes state and its dispositions change as described above.

Clearly P outputs the unencoded form of a permutation generated by A and the proof is complete. ∎

Proof of Theorem 8

Let X and Y be regular sets of permutations (closed or not) and let P and Q be finite state permuters for them. We construct the serial composition of P and Q in much the same way that two transducers are composed. The set of modes is the product set of the modes of P and Q. A transition of P that causes no output and a transition of Q are reflected by changes to the modes and dispositions in the appropriate component. A transition that causes output from P causes the output symbol to be moved into Q under the control of the transition rules for Q.

The new permuter is associated with the composition of P and Q and the proof is complete. ∎

3 Transposition switches in series

This section discusses a rather simple permuting machine and the rather more complex machines that can be formed by combining any number of them in series. The machine is a *transposition switch*. It is capable of swapping the members of any set of (disjoint) adjacent pairs in a list of length n. For example, given the list 1234, it could generate the permutations $1234, 2134, 1324, 1243, 2143$.

Let T be the set of permutations associated with this switch. The set T is closed, has the basis $\{231, 312, 321\}$ and is enumerated by the Fibonacci numbers (first proved in [10]). Its permutations are layered with layers of size at most 2 and

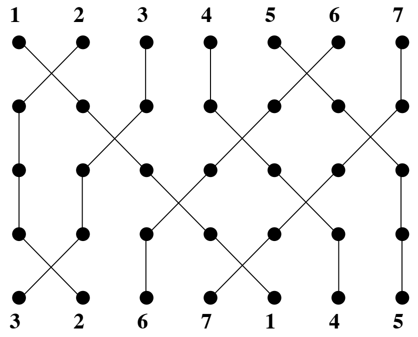

Figure 2: T^4 contains 3267145

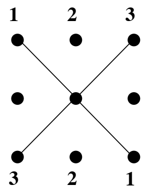

Figure 3: T^2 does not contain 321

could hardly be simpler. However, if we combine several identical transposition switches in series we quickly increase the complexity of the situation.

We may depict the way a series arrangement of transposition switches generates a permutation by a grid diagram which makes explicit which transpositions are being used at each stage. An example of a particular permutation in T^4 being generated is shown in Figure 2. We may consider such a diagram as consisting of n wires, one for each symbol. The ith wire shows the position occupied by i at each stage. Crossing wires (travelling left or right) denote a transposition at that stage; a wire that that does not cross another (travelling vertically) denotes a fixed point at that stage. Since each stage represents a permutation no two wires can go through the same grid point. The illegality of the wire crossing in the diagram in Figure 3 shows that $321 \notin T^2$.

By using the techniques given in Section 2 we shall prove that there are algorithms for all of the following problems:

1. Recognising in time $O(n)$ when a permutation of length n lies in T^k
2. Enumerating the permutations of T^k

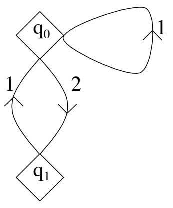

Figure 4: T is regular

3. Finding the basis of T^k (even if it is infinite).

The key to all these results is the observation that T is a regular set of permutations defined by the finite state machine in Figure 4. The reason for this is that each permutation of T encodes as a word over the alphabet $\{1, 2\}$ with no two consecutive 2s. This is exactly the language defined by the illustrated machine.

Theorem 8 tells us that T^k is also regular. Furthermore, as the proof of this theorem is constructive, it produces a finite state automaton that recognises T^k. This automaton is the linear time recogniser that justifies the first statement above and then, by using standard techniques from automata theory, we can enumerate the permutations of T^k by means of a rational generating function. To find the basis of T^k we use Corollary 1 of [1] which gives a method for constructing it as a regular expression. Unfortunately all these constructions have very high complexity in terms of k and our implementations are not effective beyond $k = 7$. Therefore *ad hoc* arguments are often simpler such as the one given in the following result.

Proposition 11 *For all $k \geq 5$, T^k is not finitely based.*

Proof: We shall consider the case $k = 5$ only since higher values are argued in the same way. Consider the following set of partially specified permutations of length $4i + 5$ $(i = 1, 2, \ldots)$

4	6	8	10	12	14	16	18	$\ldots$	$4i-4$	$4i-2$	$4i$	$4i+2$
8	1	12	6	16	10	20	14	$\ldots$	$4i$	$4i-6$	$4i+5$	$4i-2$

In this notation the only positions of the permutation that are explicitly given are those which appear in the first line. The second line gives the value of the permutation at that position. For the moment we shall not need to define the terms of the permutation at the other positions.

Irrespective of the unspecified terms these permutations do not belong to T^5. To see this we shall demonstrate that there is no wire diagram to represent such

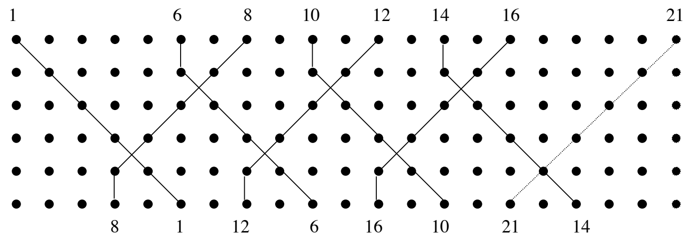

Figure 5: A series of forced wire positions and a final impossible wire

a permutation in 5 stages. If such a diagram existed then, because symbol 1 occurs in position 6 there would have to be a wire that carried symbol 1 left by one position in each of the 5 stages. Now consider symbol 8 which occurs in position 4. Its wire must have moved leftwards 4 times and remained in the same place once. However, as this wire and the wire carrying 1 cannot intersect on a grid point, its stationary point must be at the fifth stage. Thus this wire, like the wire carrying 1, is forced to be positioned in one way only. Next we argue that the symbol 6, in position 10, has had to move to the right by 4 positions which entails 4 stages in which it moves right by one step and one stage in which it is stationary. However, in order not to intersect the wire carrying 8, the stationary stage must occur in the first stage. So the positioning of this wire is also forced. We continue to argue in this way for the wires carrying successively $12, 10, 16, 14, \ldots$ until we consider the symbol $4i + 5$ in position $4i$ (see Figure 5 for the case $i = 4$). At this point we reach a contradiction since this symbol has to move left in all 5 stages and its wire then intersects with the wire for $4i - 2$.

We have now displayed infinitely many permutations not belonging to T^5. To actually find infinitely many basis elements we must now specify the entire permutation in all cases. We do this in the simplest way possible: the remaining terms of the permutation are arranged in increasing order of size and then allocated serially to the unallocated positions. Thus for the permutation arising from Figure 5, where $i = 4$, we would obtain

2 3 4 8 5 1 7 12 9 6 11 16 13 10 15 21 17 14 18 19 20

A rather laborious check now confirms that, if any symbol is omitted from such a permutation, the result lies in T^5. ∎

For the values $k \leq 4$, our calculations show that T^k is finitely based. It turns out that the basis elements of T^2 have length at most 4, for T^3 their length is at most 5, and for T^4 their length is at most 8. These calculations, whilst computationally intensive, all rely on the regular set methods of [1]. These methods also apply for higher values of k although, in practice, we cannot go beyond $k = 7$. As an example of what they can achieve we record the fact that

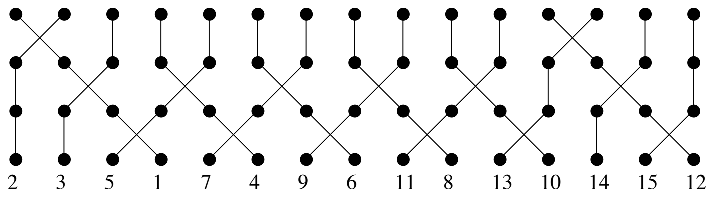

Figure 6: 2 3 5 1 7 4 9 6 11 8 13 10 14 15 12 $\in T^3$

there are 188022304133598393331500 elements of length 50 in the basis of T^7.

One further respect in which the structure of T^k becomes more complex as k increases is given in the next result.

Proposition 12 *T^k is partially well-ordered if and only if $k \leq 2$.*

Proof: Since $T^{k-1} \subseteq T^k$ it suffices to prove that T^3 has an infinite antichain whilst T^2 is partially well-ordered.

The infinite set of permutations $2\ 3\ 5\ 1\ 7\ 4\ \cdots 2n+1\ 2n-2\ 2n+2\ 2n+3\ 2n$ $(n \geq 2)$ is an infinite antichain (a minor variant on the antichain appearing in [11]). But each permutation lies in T^3 as exemplified in Figure 6; the pattern of this diagram obviously generalises.

To prove that T^2 is partially well-ordered it suffices (by a result in [4]) to prove that its set of indecomposables is partially well-ordered. For an indecomposable permutation σ we consider the cycle containing symbol 1. As each symbol differs by 1 or 2 from the previous symbol in the cycle it must have one of the forms

$$(1, 3, 5, \ldots, 2m-1, 2m, 2m-2, 2m-4, \ldots, 4, 2)$$

or

$$(1, 3, 5, \ldots, 2m-1, 2m+1, 2m, 2m-2, \ldots, 4, 2)$$

or the inverse of such a cycle.

In all cases the cycle contains a contiguous set of terms and, since σ is indecomposable, it must be the whole of σ. The former cycle is clearly involved in the latter one. Therefore the indecomposable permutations fall into two ascending chains and, as such, obviously form a partially well-ordered set. ■

The last result shows rather more about T^2. It also shows that the numbers of indecomposable permutations of lengths $1, 2, 3, \ldots$ is $1, 1, 2, 2, \ldots$. Hence the generating function of T^2 is $\frac{1}{1-g(x)}$ where $g(x) = x + x^2 + 2x^3/(1-x)$. This comes to

$$\frac{1-x}{1-2x-x^3}$$

The generating functions for some higher powers of T can be calculated using the automata-theoretic methods. For example, the generating function for T^3 is

$$\frac{1 - x - 2x^2 + x^3 - x^4}{1 - 2x - 2x^2 - 6x^4 + 2x^5 - x^6}$$

4 Conclusions and open questions

We have shown that serial combinations of permuting machines are closely connected with compositions of pattern restricted sets. We have also given a number of general families of closed sets whose compositions can be computed. However, it is an apparently difficult question to decide of two finitely based closed sets whether their composition is finitely based and we have seen examples of rather simple finitely based closed sets whose composition is not finitely based ($\mathcal{S} \circ \mathcal{S}, T^2 \circ T^3$). Beyond this our knowledge is extremely sketchy.

Suppose that C is the set of all powers of all cycles $(1, 2, \ldots, n)$. Thus $C \circ X$ consists of the permutations in X and all their cyclic variants. We expect the following conjecture to be true.

Conjecture *If X is finitely based then $C \circ X$ is finitely based,*

We have verified this conjecture in the two cases that X has the basis $\{321\}$ or the basis $\{231\}$.

Our results on transposition switches in series show that, for each fixed k, there is a linear time algorithm to decide whether a permutation σ belongs to T^k. The algorithms are derived from regular encodings and finite automata. However, we have not given a uniform treatment in the sense that, when k is not specified in advance, we can say what exchanges should be applied to sort σ, especially in the first round. An answer to the following problem would greatly enhance our understanding of transposition switches.

Problem *Find an algorithm that can sort a given permutation in the smallest number of applications of a transposition switch.*

References

[1] M. H. Albert, M. D. Atkinson, N. Ruškuc: Regular closed sets of permutations, Theoretical Computer Science 306 (2003), 85–100.

[2] M. D. Atkinson: Restricted Permutations, Discrete Math. 195 (1999), 27–38.

[3] M. D. Atkinson, R. Beals: Permuting mechanisms and closed classes of permutations, Australian Computer Science Communications 21 (3) 119–127.

[4] M. D. Atkinson, M. M. Murphy, N. Ruškuc: Partially well-ordered closed sets of permutations, Order 19 (2002) 101–113.

[5] M. D. Atkinson, D. Tulley, M. J. Livesey: Permutations generated by token passing in graphs, Theoretical Computer Science 178 (1997), 103–118.

[6] M. Bóna: A Survey of Stack-Sorting Disciplines, Electronic J. Combin. 9(2) (2003), Paper A1.

[7] D. E. Knuth: *Fundamental Algorithms: Volume 1 of The Art of Computer Programming*, First edition, Addison-Wesley (Reading, Mass.) 1968.

[8] M. M. Murphy: *Restricted permutations, Antichains, Atomic Classes and Stack Sorting*, PhD thesis, St Andrews 2003.

[9] V. R. Pratt: Computing permutations with double-ended queues, parallel stacks and parallel queues, Proc. ACM Symp. Theory of Computing 5 (1973), 268–277.

[10] R. Simion, F. W. Schmidt: Restricted permutations, Europ. J. Combinatorics 6 (1985), 383–406.

[11] D. A. Spielman, M. Bóna: An Infinite Antichain of Permutations, Note N2, Electronic. J. Combin. 7(1), 2000.

[12] R. E. Tarjan: Sorting using networks of queues and stacks, Journal of the ACM 19 (1972), 341–346.