ECO-based Gray codes generation for particular classes of words

Extended abstract

Vincent Vajnovszki
LE2I, Université de Bourgogne
BP 47870, 21078 Dijon Cedex, France
vvajnov@u-bourgogne.fr

June 3, 2012

1 Introduction

A general Gray code for a very large family of combinatorial objects is given in [4]; it is based on generating trees (or ECO operators) and objects are encoded by their corresponding path in the generating tree and often it is possible to translate the obtained codes into codes for objects. Later, in [3, 7, 18] the ECO method is applied for generating Gray codes for some restricted classes of permutations. Motivated by these papers, we investigate the related problem for several classes of words. More precisely, we express in terms of ECO method known Gray codes for Dyck words, Grand Dyck words and combinations, and develop new Gray codes for Motzkin and Schröder words. The techniques we present here is, in a way, complementary to that of [4]: it is less general but easy to describe, understand, implement and manipulate. Also, our generating trees-based construction is an alternative to the *reversing sublists* technique [12], *shuffle combinatorial objects* [15], *order relations* [17, 19] or *bubble languages* [13], and has the advantage that intermediate produced objects are still (smaller size) objects in the same class. Some preliminary results concerning binary words have been presented in [16].

ECO operators and rules. We recall the definition of an *ECO operator* as it is formalized in [1]. Let $\mathcal{O}$ be a class of combinatorial objects and $\mathcal{O}_n$ the subclass of objects of size n. An operator ϑ on the class $\mathcal{O}$ is a family of functions (one for each n) $\vartheta : \mathcal{O}_n \to 2^{\mathcal{O}_{n+1}}$ with $2^{\mathcal{O}_n}$ being the power set of $\mathcal{O}_n$. If the operator ϑ satisfies the following conditions:

1. if $x_1, x_2 \in \mathcal{O}$, and $x_1 \neq x_2$, then $\vartheta(x_1) \cap \vartheta(x_2) = \varnothing$,

2. for each $y \in \mathcal{O}_n$, $n \geq 1$, there exists a unique $x \in \mathcal{O}_{n-1}$ such that $y \in \vartheta(x)$,

then $\{\vartheta(x)\}_{x \in \mathcal{O}_{n-1}}$, $n \geq 1$, is a partition of $\mathcal{O}_n$ and ϑ is called an ECO operator. Alternatively, an ECO operator can be expressed in terms of ECO (or succession) rules.

A *colored integer* is an integer or an integer with a subscript which is called *color*. For a colored integer e, $|e|$ denotes the value of e regardless its color and $|e| = e$ if e is simply an integer. For instance $|2_a| = 2$ and $|3| = 3$.

A *succession rule* on a set of colored integers Σ is a formal system consisting of a root $e_0 \in \Sigma$ and a set of *productions* of the form

$$\{(k) \rightsquigarrow (e_1(k))(e_2(k)) \cdots (e_{|k|}(k))\}_{k \in \Sigma}$$

with each $e_i(k) \in \Sigma$, $1 \leq i \leq |k|$, which explain how to derive, for any given $k \in \Sigma$, its $|k|$ successors, $(e_1(k)), (e_2(k)), \ldots, (e_{|k|}(k))$, see for instance [1]. In this context Σ is called the set of *labels*.

A *generating tree* induced by a succession rule is an infinite tree with the root (at level zero) labelled by (e_0). Each node labelled by (k) has $|k|$ successors with the labels given by the production rules. A tree is a generating tree for a class of combinatorial objects if there exists a bijection between the objects of size n and the nodes at level n in the tree. Note that a class of combinatorial objects may have several generating trees.

In the context of Gray codes we associate to each colored integer $k \in \Sigma$ a direction, *down* or *up*, and we denote by $\overline{k}$ the colored integer k with direction *down* and simply by k the colored integer k with direction *up*. The list of successors of $(\overline{k})$ is obtained by reversing the list of successors of (k) then reversing the direction of each element of the list. For instance, if the successors of (3) are, in order, (2), $(\overline{3})$ and $(\overline{4})$, then the successors of $(\overline{3})$ are (4), (3) and $(\overline{2})$.

Dyck words and Grand Dyck words. A *Dyck word* is a binary word with the same number of 1's and 0's and satisfying the *suffix property*: any suffix has at least as many 0's as 1's. Dyck words code a wide variety of combinatorial objects including binary trees or lattice paths. We denote by D_{2n} the set of length $2n$ Dyck words. A natural generalization of Dyck words are *p-ary Dyck words* which are binary words with exactly $p-1$ times as many 0's as 1's and satisfying the *p-th order suffix property*: any suffix has at least $p-1$ times as many 0's as 1's. We denote by D^p_{np} the set of p-ary Dyck words of length np, and $D^2_{2n} = D_{2n}$.

If the suffix property condition is dropped from the definition of Dyck (*resp.* p-ary Dyck) words, then the obtained words are called *Grand Dyck* (*resp. p-ary Grand Dyck*) *words* and the set of length $2n$ Grand Dyck (*resp.* length np p-ary Grand Dyck) words is denoted by GD_{2n} (*resp.* GD^p_{np}), and $GD^2_{2n} = GD_{2n}$. Finally, the set of length n binary words with exactly m occurrences of 0 is denoted by $C_{n,m}$ and these words code $(n-m)$-combinations of an n-element set. Obviously, $D^p_{np} \subset GD^p_{np} = C_{np,n(p-1)}$.

Motzkin and Schröder words. A *Motzkin word* is a word over the alphabet $\{0, 1, a\}$ which after erasing each occurrence of a gives a Dyck word; and we denote M_n the set of length n Motzkin words. A *Schröder word* is a Motzkin word in which each length maximal factor of the form $aa \dots a$ has even length. Clearly Schröder words have even length and S_{2n} denotes the set of length $2n$ Schröder words.

Gray codes and generating algorithms. A *Gray code* is an infinite collection of word-lists, one list for words with same length, such that the number of positions in which two consecutive words in each list differ is bounded (independently of the word-length) [20]. For a $\delta \geq 1$, a Gray code has *distance* δ if consecutive words differ in at most δ positions. In particular, a Gray code for D^p_{np} (for GD^p_{np}, or for $C_{n,m}$) is an exhaustive list for the binary words under consideration such that successive words differ by the transposition of a 1 and a 0, that is the list is 2-Gray code; a Gray code is *circular* if the last and the first word in the list differ in the same way, and a Gray code is called *homogeneous* if the 1 and the 0 that exchange positions are separated only by 0's.

In the context of ECO-based Gray codes, a (k) or $(\overline{k})$ labeled node in the generating tree corresponds to a word with k successors; and the successors of a $(\overline{k})$ labeled node are the same as for (k), but in reverse order.

An algorithm for generating a list of words is called *CAT* [14] (as Constant Average Time) if the number of operations necessary to transform each word into its successor in the list, is constant in average.

2 Dyck words

Let $d = d_1 d_2 \dots d_{np}$ be a p-ary Dyck word of length np, and ℓ be its length-maximal 0's suffix. In other words, $d = d_1 d_2 \dots d_{np-\ell} 0^\ell$ with $d_{np-\ell} = 1$, and the suffix 0^ℓ is called the *last descent* of d. For each u, $0 \leq u \leq \ell$, the word $d_1 d_2 \dots d_{np-\ell} 0^u 1 0^{\ell-u+p-1}$ is a p-ary Dyck word of length $(n+1)p$ called a *successor* of d. This succession rule is called *last descent rule*. For example, the three successors of the length six binary Dyck word 110100 according to this rule are: 11011**0**0**0**, 1101010**0** and 11010010; see Fig. 1 (a) where Dyck words are represented as lattice paths.

A (k) labeled node in the generating tree corresponds to a p-ary Dyck word with its last descent of length $k-1$. Its (ordered list of) successors are obtained by inserting, from right to left, a 1 into its last descent, then adding a 0^{p-1} suffix; and the successors of a $(\overline{k})$ labeled node are the same but in reverse order. For example, the Dyck word 110100 in Fig. 1 (a) must be labeled by $(\overline{3})$ in the generating tree.

Theorem 1. *The succession rule*

$$\left\{ \begin{array}{l} (1) \\ (k) \rightsquigarrow (p)(\overline{p+1}) \cdots (\overline{p+k-1}) \end{array} \right. \tag{1}$$

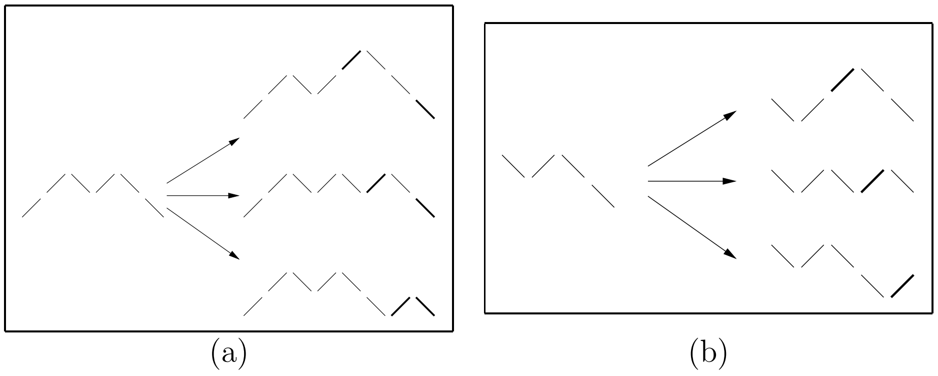

Figure 1: (a) The three successors, according to the last descent rule, of the binary Dyck word 110100: 1101**1**00**0**, 110101**0**0 and 1101001**0**. (b) The three successors, according to the last descent rule, of the binary word 0100: 01**1**00, 010**1**0 and 0100**1**.

gives a circular Gray code for p-ary Dyck words, where the root of the generating tree is the empty word ϵ.

See Fig. 2 for the first levels of the generating tree induced by the succession rule (1) with $p = 2$; and Fig. 3 (a) for the first four levels of the 3-ary Dyck words ($p = 4$) generated by this rule.

Proposition 1. *For $p \geq 2$, the first and last length np word given by the succession rule (1) is $(10^{p-1})^n$, $n \geq 1$, and $110^{2p-2}(10^{p-1})^{n-2}$, $n \geq 2$, respectively.*

Remark 1. *For $p = 2$, the Gray code induced on D_{2n} by the succession rule (1) is the reverse of Ruskey-Proskurowski's Gray code [11].*

Now, we give a succession rule for a more restrictive Gray code for D^p_{np}; as expected, this rule is more complicated and involves colored labels.

Theorem 2. *The succession rule*

$$\begin{cases} (1)_a \\ (k_a) \rightsquigarrow (p_a)\ (\overline{p+1})_b\ (\overline{p+2})_b\ \cdots (\overline{p+k-1})_b \\ (k_b) \rightsquigarrow (\overline{p+k-1})_a\ (p_a)\ (\overline{p+1})_b\ (\overline{p+2})_b\ \cdots\ (\overline{p+k-2})_b \end{cases} \tag{2}$$

gives a homogeneous Gray code for p-ary Dyck words, where the root of the generating tree is the empty word ϵ.

Proposition 2. *The first and last length np word given by the succession rule (2) is $(10^{p-1})^n$ and $1^n0^{(p-1)n}$, respectively, for $n \geq 1$.*

Remark 2. *The Gray code induced on D^p_{np} by the succession rule (2) is Eades-McKay-Bultena-Ruskey's Gray code [8, 5].*

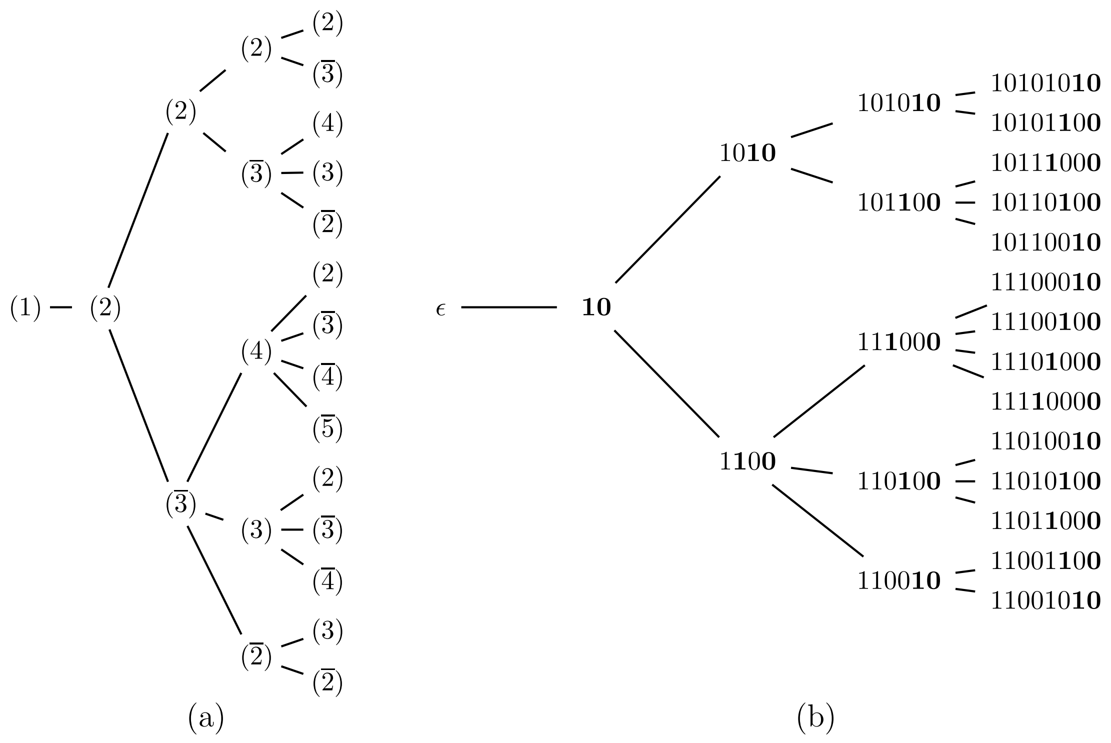

Figure 2: (a) The first five levels of the tree induced by the succession rule (1) for $p = 2$; and (b) the corresponding generated binary Dyck words.

3 Combinations and Grand Dyck words

Let $d = d_1 d_2 \dots d_n$ be a binary word in $C_{n,m}$, and ℓ be its length-maximal 0's suffix; that is, $d = d_1 d_2 \dots d_{n-\ell} 0^\ell$ with $d_{n-\ell} = 1$, and as above, the suffix 0^ℓ is called the *last descent* of d. For each u, $0 \le u \le \ell$, the word $d_1 d_2 \dots d_{n-\ell} 0^u 1 0^{\ell - u}$ is a length $(n+1)$ binary word in $C_{n+1,m}$, called a successor of d. This succession rule is the *last descent rule* for combinations. For example, the three successors of the length four binary word 0100 are: 01**1**00, 010**1**0 and 0100**1**; see Fig. 1 (b) where binary words are represented as lattice paths. This recursive construction of $C_{n,m}$ gives the following succession rule, where the size zero object is 0^m, the unique binary word in $C_{m,m}$.

Theorem 3. *For a fixed $m \ge 1$, the succession rule*

$$\left\{ \begin{array}{l} (m+1) \\ (k) \rightsquigarrow (1)(\overline{2}) \cdots (\overline{k}) \end{array} \right. \tag{3}$$

gives a (circular) Gray code for $C_{n,m}$, $n \ge m$.

Proposition 3. *The first and last length n word given by the succession rule (3) are $0^m 1^{n-m}$, and $10^m 1^{n-m-1}$, respectively.*

See Fig. 3 (b) for the first five levels of the generating tree induced by the succession rule (3) for $m = 3$. In particular, for a given $p \ge 2$, when $n = \frac{m}{p-1} \cdot p$, (that is, at level $n - m$ in the generating tree) the succession rule (3) yields a Gray code for $GD_n^p = C_{n,m}$.

Remark 3. *The Gray code induced on $C_{n,m}$ by the succession rule (3) is the* revolving door *Gray code [9, 10].*

4 Motzkin words

Let $w \in M_n$ be a length-n Motzkin word, s its length-maximal suffix which does not contain the letter 1, and $k = |s|_a + 1$. The following points below give the successors of w, and so an ECO operator for the set of Motzkin words.

- The word wa is a Motzkin word of length $n+1$ with $k+1$ successors;
- If $|s|_a > 0$, then for a given j, $|s|_a \ge j > 0$ let $w'as''$ be the factorization of w where s'' is the suffix of w with exactly $j-1$ occurrences of a. The word $v = w'1s''0$ is a Motzkin word of length $n+1$; it has j successors.

Theorem 4. *The succession rule*

$$\left\{ \begin{array}{l} (1) \\ (k) \rightsquigarrow (\overline{1})(\overline{2}) \cdots (\overline{k-1})(k+1) \end{array} \right. \tag{4}$$

gives a Gray code for M_n with distance 4.

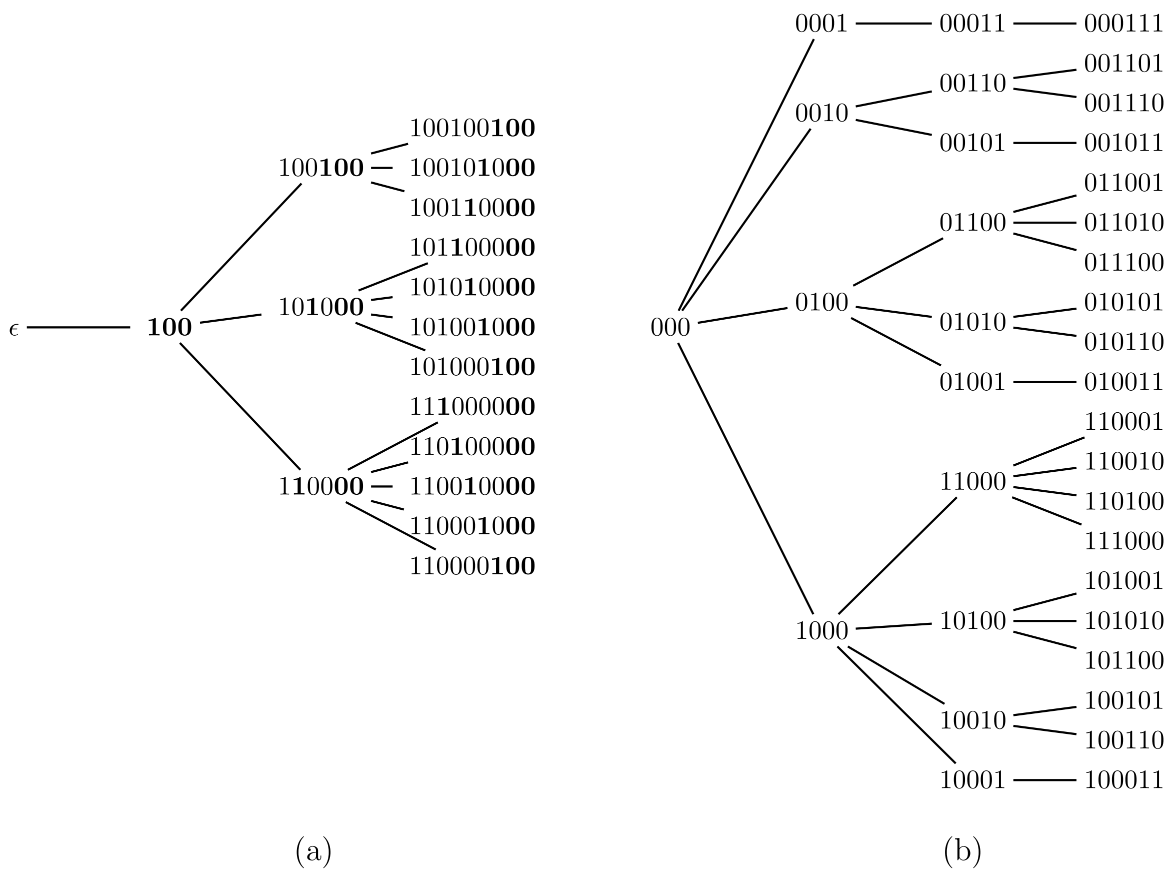

Figure 3: (a) The first four levels of the generated 3-ary Dyck words ($p = 3$) induced by the succession rule (1). (b) The first four levels of the generating tree induced by the succession rule (3) with $m = 3$.

5 Schröder words

Let $w \in S_{2n}$, and k be the length of its length-maximal 0's suffix, plus one; and w' the prefix of w such that $w = w'0^{k-1}$. Then w has $2k$ successors and the following points below give the successors of w, and so an ECO operator for the set of Schröder words.

- $waa \in S_{2n+2}$ is a length $2n+2$ Schröder word with 2 successors;
- $w'100^{k-1} \in S_{2n+2}$ is a length $2n+2$ Schröder word with $2k+2$ successors.

In addition, if $k > 1$, then for any j, $1 \leq j \leq k-1$ we have

- $w'0^{k-1-j}aa0^j \in S_{2n+2}$; it has $2j$ successors;
- $w'0^{k-j}100^{j-1} \in S_{2n+2}$; it has $2j$ successors.

Theorem 5. *The succession rule*

$$\begin{cases} (2) \\ (2k) \rightsquigarrow (2)(\overline{4})(\overline{4})(\overline{6})(\overline{6}) \cdots (\overline{2k})(\overline{2k})(\overline{2k+2}) \end{cases} \tag{5}$$

gives a Gray code for S_n with distance 5.

6 Algorithmic implementation

The direct algorithmic implementation of the succession rule (1) gives the algorithm `gen_Dyck_up` below. The words are stored in the global array d, and for convenience it is initialized by 0^{np}, and the main call is `gen_Dyck_up(0,1)` corresponding to the root of the generating tree. The procedure `gen_Dyck_down`, not shown, essentially executes the statements of `gen_Dyck_up` in reverse order and replaces the calls of `gen_Dyck_up` by `gen_Dyck_down` and vice-versa.

```
procedure gen_Dyck_up($size$, $k$)
local $i$;
if $size = n \cdot p$ then Print($d$);
else$d[size+1] := 1$;
    gen_Dyck_up($size+p$, $p$);
    $d[size+1] := 0$;
    for $i$ from $p+1$ to $k+p-1$ do
        $d[size+p+1-i] := 1$;
        gen_Dyck_down($size+p$, $i$);
        $d[size+p+1-i] := 0$;
    end do
end if
end procedure.
```

Proposition 4. *Procedure* `gen_Dyck_up` *generates p-ary Dyck words of length np in constant average time.*

Proof. This algorithm satisfies the following properties:

1. the total amount of computation in each call is proportional with the number of direct calls produced by this call,

2. each non-terminal call, except the root, produces at least two recursive calls (i.e., there is no call of degree one, except to the main call), and

3. each terminal call (degree-zero call) produces a new permutation.

In [14] it is shown that a generating algorithm satisfying these properties runs in constant average time. See Figure 2 (b) for the first levels of the tree induced by the call of `gen_Dyck_up(0,1)` with $p = 2$. □

The algorithmic implementation of the succession rule (2) is similar to the one of (1) and we have:

Proposition 5. *The algorithmic implementation of the succession rule (2) gives a CAT generating algorithm for a homogeneous Gray code for p-ary Dyck words of length np.*

Generally, the implementation of the succession rule (3) for $C_{n,m}$ does not give a CAT algorithm. Nevertheless, for an integer $p \geq 2$ and for $n = \frac{mp}{p-1}$, numerical evidences show that $\frac{\textit{total amount of computation}}{\textit{number of generated words}} \leq 2$; and so, it seems that the succession rule (3) yields a CAT algorithm for the set of p-ary Grand Dyck words.

Based on succession rules (4) and (5), similar algorithms with same time complexity can be developed for Motzkin and Schröder words.

References

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

[2] S. Bacchelli, E. Barcucci, E. Grazzini, E. Pergola, Exhaustive generation of combinatorial objects by ECO, *Acta Informatica*, **40** (8) (2004), 585-602.

[3] J-L. Baril and P-T. Do, ECO-generation for p-generalized Fibonacci and Lucas permutations, *Journal of Pure Mathematics and Applications*, **17** (1-2) (2006), 19-37.

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

[5] B. Bultena, and F. Ruskey, An Eades–McKay algorithm for well–formed parentheses strings, *Information Processing Letters* **68** (1998), 255–259.

[6] A. Del Longo, A. Frosini, S. Rinaldi. ECO method and the exhaustive generation of convex polyominoes, *Discrete Mathematics and Theoretical Computer Science, LNCS 2731*, 2003, 129–140.

[7] W.M.B. Dukes, M.F. Flanagan, T. Mansour, V. Vajnovszki, Combinatorial Gray codes for classes of pattern avoiding permutations, *TCS* **396** (2008), 35-49.

[8] G. Eades, B. McKay, An algorithm for generating subsets of fixed size with a strong minimal change property, *IPL* **19** (1984), 131–133.

[9] C.N. Liu, D.T. Tang. Algorithm 452, enumerating m out of n objects. *Comm. ACM* **16** (1973), 485.

[10] A. Nijenhuis, H.S. Wilf. *Combinatorial Algorithms for Computers and Calculators.* Academic Press, 1978.

[11] F. Ruskey, A. Proskurowski, Generating binary trees by transpositions, *Journal of Algorithms* **11** (1990), 68–84.

[12] F. Ruskey, Simple combinatorial Gray codes constructed by reversing sublists, in *ISAAC Conference, LNCS*, **762** (1993), 201–208.

[13] F. Ruskey, Joe Sawada, Aaron Williams, Binary bubble languages and cool-lex order, to appear in Journal of Combinatorial Theory, Series A.

[14] F. Ruskey, Combinatorial Generation, book in preparation.

[15] V. Vajnovszki, Gray visiting Motzkins, *Acta Informatica* 38(2002), 793-811.

[16] V. Vajnovszki, Simple Gray codes constructed by ECO method, *JMIT*, Mons, Belgique, 27-30 august 2008.

[17] V. Vajnovszki, More restrictive Gray codes for necklaces and Lyndon words, *Information Processin Letters*, **106** (3), (2008) 96-99.

[18] V. Vajnovszki, Generating involutions, derangements, and relatives by ECO, *DMTCS*, **12** (1) (2010) 109-122.

[19] V. Vajnovszki, R. Vernay, Restricted compositions and permutations: from old to new Gray codes, *Information Processin Letters*, **111** (2011), 650-655.

[20] T. Walsh, Generating Gray codes in $O(1)$ worst-case time per word, $4th$ Discrete Mathematics and Theoretical Computer Science Conference, Dijon-France, 7-12 July 2003 (*LNCS* 2731, 71-88).

[21] M. Weston, V. Vajnovszki, Gray codes for necklaces and Lyndon words of arbitrary base, *PuMA*, **17** (1-2) (2006), 175–182.