Two Reflected Gray Code based orders on some restricted growth sequences

Ahmad Sabri and Vincent Vajnovszki
LE2I, Université de Bourgogne
BP 47870, 21078 Dijon Cedex, France
{ahmad.sabri}{vvajnov}@u-bourgogne.fr

January 23, 2014

Abstract

We consider two order relations: that induced by the m-ary reflected Gray code and a suffix partitioned variation of it. We show that both of them when applied to some sets of restricted growth sequences still yield Gray codes. These sets of sequences are: subexcedant and ascent sequences, restricted growth functions and staircase words. In particular, we give the first suffix partitioned Gray codes for restricted growth functions and ascent sequences; these latter sequences code various combinatorial classes as interval orders, upper triangular matrices without zero rows and zero columns whose non-negative integer entries sum up to n, and certain pattern-avoiding permutations. For each Gray code we give efficient exhaustive generating algorithms and compare the obtained results.

1 Introduction and motivations

The term "Gray code" was taken from Frank Gray, who patented *Binary Reflected Gray Code* (BRGC) in 1953 [1]. The concept of BRGC is extended to *Reflected Gray Code* (RGC), to accommodate m-ary sequences, with $m > 2$ [2]. In these Gray codes, successive sequence differ in a single position, and by $+1$ or -1 in this position. More generally, if a list of sequences is such that the *Hamming distance* between successive sequences (the number of positions in which the sequences differ) is bounded from above by a constant d, then the list is said to be a *d-Gray code*. So, in particular, BRGC and RGC are 1-Gray codes. In addition, if the positions where the successive sequences differ are adjacent, then we say that the list is a *d-adjacent* Gray code.

For a long time, the design of Gray codes for combinatorial classes and their corresponding generating algorithms was an ad-hoc task, that is, done case by case according to the class under consideration. Recently, general techniques which fit to large classes of combinatorial objects were developed and used. Among them are, for example, *reflectable languages* [3] (see also [4] for *defining sequences* and [5] for *stability property* techniques), *prefix rotations* (yielding *bubble languages*, see [6, 7] and references therein), and Reflected Gray Code based order relations; this last technique was used implicitly, for example in [8, 9], and developed systematically as a general method in order to define Gray codes (and corresponding generating algorithms) for various classes as Fibonacci and Lucas strings, restricted compositions, Lyndon words and relatives [10–14]. The results presented in this paper are in the light of this last direction. More precisely, we show that two order relations induced by Reflected Gray Code and its variation also give Gray codes for some classes of restricted growth sequences defined by means of statistics. These classes are: subexcedant and ascent sequences, restricted growth functions, and staircase words. We give efficient (CAT) generating algorithm for each obtained Gray code.

For the combinatorial classes we consider in this paper, there already exist Gray codes for them, see for instance [15–17]. The novelty of those presented here consists in that their definitions are put under the same roof: RGC inspired order relations. As a consequence of these constructions it follows that if U and V are two such classes, and $U \subset V$, then the Gray code for U is a (possibly scattered) sublist of that of V. Moreover, some applications may require suffix partitioned Gray code lists, and in this case general techniques as reflectable languages can not be applied in order to define Gray codes. Indeed, some suffixes of ascent sequences, restricted growth functions or staircase words are not extendible in at least two different ways as required by this technique, and the codes we present in Section 4 are suffix partitioned. Additionally, our generating algorithms for suffix partitioned lists are more appropriate for large-scale parallelization.

2 Preliminaries

2.1 Gray code orders

Let $G_n(m)$ be the set of length n m-ary sequences $s_1s_2\ldots s_n$ with $s_i \in \{0,1,\ldots,m-1\}$; clearly, $G_n(m)$ is the product set $\{0,1,\ldots,m-1\}^n$. The *Reflected Gray Code* (*RGC* for short) for the set $G_n(m)$, denoted by $\mathcal{G}_n(m)$, is the natural extension of the Binary Reflected Gray Code to this set. The list $\mathcal{G}_n(m)$ is defined recursively by the following relation [2]:

$$\mathcal{G}_n(m) = \begin{cases} \epsilon & \text{if} \quad n=0, \\ 0\mathcal{G}_{n-1}(m),\ 1\overline{\mathcal{G}_{n-1}(m)},\ 2\mathcal{G}_{n-1}(m),\ \ldots,\ (m-1)\mathcal{G}'_{n-1}(m) & \text{if} \quad n>0, \end{cases} \tag{1}$$

where ϵ is the empty sequence, $\overline{\mathcal{G}_{n-1}(m)}$ is the reverse of $\mathcal{G}_{n-1}(m)$, and $\mathcal{G}'_{n-1}(m)$ is $\mathcal{G}_{n-1}(m)$ or $\overline{\mathcal{G}_{n-1}(m)}$ according to m is odd or even.

In $\mathcal{G}_n(m)$, two successive sequences differ in a single position and by $+1$ or -1 in this position. A list for a set of sequences induces an order relation to this set, and we give two order relations induced by the RGC and its variation, namely RGC order [10] and Co-RGC order.

We adopt the convention that lower case bold letters represent tuples, for example: $\boldsymbol{s} = s_1s_2\ldots s_n$, $\boldsymbol{a} = a_1a_2\ldots a_k$, and $\boldsymbol{b} = b_{k+1}b_{k+2}\ldots b_n$.

Definition 1. The *Reflected Gray Code order* $\prec$ on $G_n(m)$ is defined as: $\boldsymbol{s} = s_1s_2\ldots s_n$ is less than $\boldsymbol{t} = t_1t_2\ldots t_n$, denoted by $\boldsymbol{s} \prec \boldsymbol{t}$, if either

- $\sum_{i=1}^{k-1} s_i$ is even and $s_k < t_k$, or
- $\sum_{i=1}^{k-1} s_i$ is odd and $s_k > t_k$,

where k is the leftmost position where $\boldsymbol{s}$ and $\boldsymbol{t}$ differ.

It is easy to see that $\mathcal{G}_n(m)$ defined in relation (1) lists sequences in $G_n(m)$ in $\prec$ order.

Now we give a variation of $\mathcal{G}_n(m)$. Let $s_1s_2\ldots s_n$ be a sequence in $G_n(m)$. The complement of s_i, $1 \le i \le n$, is

$$(m-1-s_i),$$

and the reverse of $s_1 s_2 \dots s_n$ is

$$s_n s_{n-1} \dots s_1.$$

Let $\widetilde{\mathcal{G}}_n(m)$ be the list obtained by transforming each sequence $\boldsymbol{s}$ in $\mathcal{G}_n(m)$ as follows:

- complementing each digit in $\boldsymbol{s}$ if m is even, or complementing only digits in odd positions if m is odd, then
- reversing the obtained sequence.

Clearly, $\widetilde{\mathcal{G}}_n(m)$ is also a Gray code for $G_n(m)$, and the sequences therein are listed in Co-Reflected Gray Code order, as defined formally below.

Definition 2. The *Co-Reflected Gray Code order* $\prec_c$ on $G_n(m)$ is defined as:
$\boldsymbol{s} = s_1 s_2 \dots s_n$ is less than $\boldsymbol{t} = t_1 t_2 \dots t_n$, denoted by $\boldsymbol{s} \prec_c \boldsymbol{t}$, if either

- $\sum_{i=k+1}^{n} s_i + (n-k)$ is even and $s_k > t_k$, or
- $\sum_{i=k+1}^{n} s_i + (n-k)$ is odd and $s_k < t_k$,

where k is the rightmost position where $\boldsymbol{s}$ and $\boldsymbol{t}$ differ.

Although this definition sounds somewhat arbitrary, as we will see in Section 4, it turns out that $\prec_c$ order gives suffix partitioned Gray codes for some sets of restricted growth sequences. Obviously, the restriction of $\mathcal{G}_n(m)$ (resp. $\widetilde{\mathcal{G}}_n(m)$) to a set of sequences is simply the list of sequences in the set listed in $\prec$ (resp. $\prec_c$) order.

2.2 Restricted growth sequences defined by means of statistics

Through this paper we consider sequences over non-negative integers. A *statistic* on a set of sequences is an association of an integer to each sequence in the set. For a sequence $s_1 s_2 \dots s_n$, its length minus one, numbers of ascents/levels/descents, maximal value, and last value are classical examples of statistics. They are defined as follows, see also [17]:

- $\mathsf{len}(s_1 s_2 \dots s_n) = n - 1$;
- $\mathsf{asc}(s_1 s_2 \dots s_n) = \mathrm{card}\{i \,|\, 1 \le i < n \text{ and } s_i < s_{i+1}\}$;
- $\mathsf{lev}(s_1 s_2 \dots s_n) = \mathrm{card}\{i \,|\, 1 \le i < n \text{ and } s_i = s_{i+1}\}$;
- $\mathsf{des}(s_1 s_2 \dots s_n) = \mathrm{card}\{i \,|\, 1 \le i < n \text{ and } s_i > s_{i+1}\}$;
- $\mathsf{m}(s_1 s_2 \dots s_n) = \max\{s_1, s_2, \dots, s_n\}$;
- $\mathsf{lv}(s_1 s_2 \dots s_n) = s_n$.

If st is one of the statistics len, asc, m, and lv, then st satisfy the following:

$$\mathsf{st}(s_1 s_2 \dots s_n) \le n - 1, \tag{2}$$

and

$$\text{if } s_n = \mathsf{st}(s_1 s_2 \dots s_{n-1}) + 1, \text{then } s_n = \mathsf{st}(s_1 s_2 \dots s_{n-1} s_n). \tag{3}$$

On the contrary, the statistics lev and des do not satisfy relation (3). Accordingly, through this paper we will consider only the four statistics above. However, as we will point out, some of the results presented here are also true for arbitrary statistics satisfying relations (2) and (3).

Definition 3. For a given statistic st, an st-*restricted growth sequence* $s_1s_2\dots s_n$ is a sequence with $s_1 = 0$ and

$$0 \le s_{k+1} \le \mathsf{st}(s_1s_2\dots s_k) + 1 \text{ for } 1 \le k < n, \tag{4}$$

and the set of st-restricted growth sequences is the set of all sequences $s_1s_2\dots s_n$ satisfying relation (4).

From this definition, it follows that any prefix of an st-restricted growth sequence is also (a shorter) st-restricted growth sequence.

Remark 1. If st is a statistic satisfying relations (2) and (3) above, then

1. $\max\{\mathsf{st}(s_1s_2\dots s_n) \,|\, s_1s_2\dots s_n \text{ is an } \mathsf{st}\text{-restricted growth sequence}\} = n-1$;
2. if $s_1s_2\dots s_n$ is an st-restricted growth sequence, then for any k, $1 \le k < n$, $s_{k+1} = \mathsf{st}(s_1s_2\dots s_k) + 1$ implies $s_{k+1} = \mathsf{st}(s_1s_2\dots s_ks_{k+1})$.

The sets of st-restricted growth sequences, where st is one of the statistics len, asc, m, and lv, are defined below.

- The set SE_n of *subexcedant sequences* of length n is defined as:

  $$SE_n = \{s_1s_2\dots s_n \,|\, s_1 = 0 \text{ and } 0 \le s_{k+1} \le \mathsf{len}(s_1s_2\dots s_k) + 1 \text{ for } 1 \le k < n\}.$$

  SE_n is trivially in bijection with the set of length n permutations; and so, it is counted by $n!$. Notice that alternatively, $SE_n = \{0\} \times \{0,1\} \times \dots \times \{0,1,\dots,n-1\}$.

- The set A_n of *ascent sequences* of length n (originally appeared in [18] as a tool to enumerate interval orders and were extensively studied thereafter in [19–22]) is defined as:

  $$A_n = \{s_1s_2\dots s_n \,|\, s_1 = 0 \text{ and } 0 \le s_{k+1} \le \mathsf{asc}(s_1s_2\dots s_k) + 1 \text{ for } 1 \le k < n\}.$$

  A_n is in one-to-one correspondence with Stoimenow's diagrams, certain upper triangular matrices and some pattern-avoiding permutations (see Section 3.2 in [23]). The generating function for the sequence counting A_n is shown in [18] to be $\sum_{n\ge 0}\prod_{i=1}^{n}(1-(1-x)^i)$.

- The set R_n of *restricted growth functions* of length n is defined as:

  $$R_n = \{s_1s_2\dots s_n \,|\, s_1 = 0 \text{ and } 0 \le s_{k+1} \le \mathsf{m}(s_1s_2\dots s_k) + 1 \text{ for } 1 \le k < n\}.$$

  R_n is in bijection with the partitions of the set $\{1,2,\dots n\}$ (see for instance [15]) and is counted by the Bell numbers b_n, with the generating function e^{e^x-1}.

- The set S_n of *staircase words* of length n is defined as:

  $$S_n = \{s_1s_2\dots s_n \,|\, s_1 = 0 \text{ and } 0 \le s_{k+1} \le \mathsf{lv}(s_1s_2\dots s_k) + 1 \text{ for } 1 \le k < n\}.$$

  S_n is counted by Catalan numbers (see [24, exercise u, p. 222]) with the generating function $\frac{1-\sqrt{1-4x}}{2x}$.

Remark 2. $S_n \subset R_n \subset A_n \subset SE_n \subset G_n(n)$.

Below we give examples to illustrate Remark 2.

Example 1.

- *If* $\boldsymbol{s} = 010145$, *then* $\boldsymbol{s} \in SE_6$, *but* $\boldsymbol{s} \notin A_6$, $\boldsymbol{s} \notin R_6$ *and* $\boldsymbol{s} \notin S_6$.
- *If* $\boldsymbol{s} = 010103$, *then* $\boldsymbol{s} \in SE_6$, $\boldsymbol{s} \in A_6$, *but* $\boldsymbol{s} \notin R_6$ *and* $\boldsymbol{s} \notin S_6$.
- *If* $\boldsymbol{s} = 010102$, *then* $\boldsymbol{s} \in SE_6$, $\boldsymbol{s} \in A_6$, *and* $\boldsymbol{s} \in R_6$, *but* $\boldsymbol{s} \notin S_6$.
- *If* $\boldsymbol{s} = 010101$, *then* $\boldsymbol{s} \in SE_6$, $\boldsymbol{s} \in A_6$, $\boldsymbol{s} \in R_6$ *and* $\boldsymbol{s} \in S_6$.

See Table 3 for the sets S_5, R_5, and A_5 listed in $\prec$ order, and Table 4 for the same sets listed in $\prec_c$ order.
We denote by

- $\mathcal{X}_n$ the list for the set X_n in $\prec$ order, and by $\widetilde{\mathcal{X}}_n$ that in $\prec_c$ order;
- $\text{succ}_X(\boldsymbol{s})$, $\boldsymbol{s} \in X_n$, the successor of $\boldsymbol{s}$ in the set X_n listed in $\prec$ order; that is, the smallest sequence in X_n larger than $\boldsymbol{s}$ with respect to $\prec$ order;
- $\widetilde{\text{succ}}_X(\boldsymbol{s})$ the counterpart of $\text{succ}_X(\boldsymbol{s})$ with respect to $\prec_c$ order;
- $\text{first}(\mathcal{L})$ the first sequence in the list $\mathcal{L}$;
- $\text{last}(\mathcal{L})$ the last sequence in the list $\mathcal{L}$.

2.3 Constant Amortized Time algorithms and principle

An exhaustive generating algorithm is said to run in *constant amortized time* (*CAT* for short) if the total amount of computation is proportional to the number of generated objects. And so, a CAT algorithm can be considered an efficient algorithm.

Ruskey and van Baronaigien [25] introduced three CAT properties, and proved that if a recursive generating procedure satisfies them, then it runs in constant amortized time (see also [16]). They called this general technique to prove the efficiency of a generating algorithm as *CAT principle*, and the involved properties are:

1. Every call of the procedure results in the output of at least one object;
2. Excluding the computation done by the recursive calls, the amount of computation of any call is proportional to the *degree of the call*, that is, the number of call initiated by the current call;
3. The number of calls of degree one, if any, is $O(N)$, where N is the number of generated objects.

All the generating algorithms we present in this paper satisfy these three desiderata, and so they are efficient.

3 The Reflected Gray Code order for the sets SE_n, A_n, R_n, and S_n

3.1 The bound of Hamming distance between successive sequences in the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, and $\mathcal{S}_n$

Here we will show that the Hamming distance between two successive sequences in each of the mentioned lists is upper bounded by a constant, and so the lists are Gray codes.

Without another specification, X_n generically denotes one of the sets SE_n, A_n, R_n, or S_n; and $\mathcal{X}_n$ denotes its corresponding list in $\prec$ order, that is, one of the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, or $\mathcal{S}_n$. Later in this section, Theorem 1 and Proposition 2 state that, in each case, the set X_n listed in $\prec$ order yields a Gray code.

Lemma 1. *If $\boldsymbol{s} = s_1 s_2 \dots s_n$ and $\boldsymbol{t} = t_1 t_2 \dots t_n$ are two sequences in X_n with $\boldsymbol{t} = \mathrm{succ}_X(\boldsymbol{s})$ and k is the leftmost position where they differ, then $s_k = t_k + 1$ or $s_k = t_k - 1$.*

Proof. Let $\boldsymbol{t} = \mathrm{succ}_X(\boldsymbol{s})$ and k be the leftmost position where they differ. Let us suppose that $s_k < t_k$ and $s_k \neq t_k - 1$ (the case $s_k > t_k$ and $s_k \neq t_k + 1$ being similar).
It is easy to check that

$$\boldsymbol{u} = s_1 s_2 \dots s_{k-1}(s_k + 1)0 \dots 0$$

belongs to X_n, and considering the definition of $\prec$ order relation, it follows that $\boldsymbol{s} \prec \boldsymbol{u} \prec \boldsymbol{t}$, which is in contradiction with $\boldsymbol{t} = \mathrm{succ}_X(\boldsymbol{s})$, and the statement holds. □

If $\boldsymbol{a} = a_1 a_2 \dots a_k \in X_k$, then for any $n > k$, $\boldsymbol{a}$ is the prefix of at least one sequence in X_n, and we denote by $\boldsymbol{a} \,|\, \mathcal{X}_n$ the sublist of $\mathcal{X}_n$ of all sequences having the prefix $\boldsymbol{a}$. Clearly, a list in $\prec$ order for a set of sequences is a *prefix partitioned* list (all sequences with same prefix are contiguous), and for any $\boldsymbol{a} \in X_k$ and $n > k$, it follows that $\boldsymbol{a} \,|\, \mathcal{X}_n$ is a contiguous sublist of $\mathcal{X}_n$.

For a given $\boldsymbol{a} \in X_k$, the set of all x such that $\boldsymbol{a}x \in X_{k+1}$ is called the *defining set of the prefix* $\boldsymbol{a}$, and obviously $\boldsymbol{a}x$ is also a prefix of some sequences in X_n, for any $n > k$. We denote by

$$\omega_X(\boldsymbol{a}) = \max\{x \,|\, \boldsymbol{a}x \in X_{k+1}\} \tag{5}$$

the largest value in the defining set of $\boldsymbol{a}$. And if we denote $\omega_X(\boldsymbol{a})$ by M, then by Remark 1 we have

$$\begin{aligned} M &= \mathsf{st}(\boldsymbol{a}) + 1 \\ &= \mathsf{st}(\boldsymbol{a}M). \end{aligned}$$

And consequently,

$$\begin{aligned} \omega_X(\boldsymbol{a}M) &= \mathsf{st}(\boldsymbol{a}M) + 1 \\ &= M + 1. \end{aligned} \tag{6}$$

The next proposition gives the pattern of $\boldsymbol{s} \in X_n$, if $\boldsymbol{s} = \mathrm{last}(\boldsymbol{a} \,|\, \mathcal{X}_n)$ or $\boldsymbol{s} = \mathrm{first}(\boldsymbol{a} \,|\, \mathcal{X}_n)$.

Proposition 1. *Let $k < n$ and $\boldsymbol{a} = a_1 a_2 \dots a_k \in X_k$. If $\boldsymbol{s} = \mathrm{last}(\boldsymbol{a} \,|\, \mathcal{X}_n)$, then the pattern of $\boldsymbol{s}$ is given by:*

- *if $\sum_{i=1}^{k} a_i$ is odd, then $\boldsymbol{s} = \boldsymbol{a}0 \dots 0$;*
- *if $\sum_{i=1}^{k} a_i$ is even and M is odd, then $\boldsymbol{s} = \boldsymbol{a}M0 \dots 0$;*
- *if $\sum_{i=1}^{k} a_i$ is even and M is even, then $\boldsymbol{s} = \boldsymbol{a}M(M+1)0 \dots 0$;*

where M denotes $\omega_X(\boldsymbol{a})$.
Similar results hold for $\boldsymbol{s} = \mathrm{first}(\boldsymbol{a}\,|\,\mathcal{X}_n)$ by replacing 'odd' by 'even', and vice versa*, for the parity of $\sum_{i=1}^{k} a_i$.*

Proof. Let $\boldsymbol{s} = a_1a_2\dots a_k s_{k+1}\dots s_n = \mathrm{last}(\boldsymbol{a}\,|\,\mathcal{X}_n)$.
If $\sum_{i=1}^{k} a_i$ is odd, then by considering the definition of $\prec$ order, it follows that s_{k+1} is the smallest value in the defining set of $\boldsymbol{a}$, and so $s_{k+1} = 0$, and finally $\boldsymbol{s} = \boldsymbol{a}0\dots 0$, and the first point holds.
Now let us suppose that $\sum_{i=1}^{k} a_i$ is even. In this case s_{k+1} equals $\omega_X(\boldsymbol{a}) = M$, the largest value in the defining set of $\boldsymbol{a}$. When in addition M is odd, so is the summation of $\boldsymbol{a}M$, the length $k+1$ prefix of $\boldsymbol{s}$, and thus $\boldsymbol{s} = \boldsymbol{a}M0\dots 0$, and the second point holds.
Finally, when M is even, then s_{k+2} is the largest value in the defining set of $\boldsymbol{a}M$, which by relation (6) is $M+1$. In this case $M+1$ is odd, and thus $\boldsymbol{s} = \boldsymbol{a}M(M+1)0\dots 0$, and the last point holds.
The proof for the case $\boldsymbol{s} = \mathrm{first}(\boldsymbol{a}\,|\,\mathcal{X}_n)$ is similar. □

By Proposition 1 above, we have the following:

Theorem 1. *The lists $\mathcal{A}_n$, $\mathcal{R}_n$ and $\mathcal{S}_n$ are 3-adjacent Gray codes.*

Proof. Let $\mathcal{X}_n$ be one of the lists $\mathcal{A}_n$, $\mathcal{R}_n$ or $\mathcal{S}_n$, and $\boldsymbol{t} = \mathrm{succ}_X(\boldsymbol{s})$. Let k be the leftmost position where $\boldsymbol{s}$ and $\boldsymbol{t}$ differ, and let us denote by $\boldsymbol{a}$ the length k prefix of $\boldsymbol{s}$ and $\boldsymbol{a}'$ that of $\boldsymbol{t}$; so, $\boldsymbol{s} = \mathrm{last}(\boldsymbol{a}\,|\,\mathcal{X}_n)$ and $\boldsymbol{t} = \mathrm{first}(\boldsymbol{a}'\,|\,\mathcal{X}_n)$. If $k+3 \le n$, then by Proposition 1, it follows that $s_{k+3} = s_{k+4} = \dots = s_n = 0$ and $t_{k+3} = t_{k+4} = \dots = t_n = 0$. So $\boldsymbol{s}$ and $\boldsymbol{t}$ differ only in position k, and possibly in position $k+1$ and in position $k+2$.
Now we show the adjacency, that is, if $k+2 \le n$ and $s_{k+1} = t_{k+1}$ implies $s_{k+2} = t_{k+2}$. If $s_{k+1} = t_{k+1}$, by Lemma 1, it follows that the summation of the length k prefix of $\boldsymbol{s}$ and that of $\boldsymbol{t}$ have different parity, and two cases can occur:
• $s_{k+1} = t_{k+1} = 0$, and by Proposition 1, it follows that $s_{k+2} = t_{k+2} = 0$; or
• $s_{k+1} = t_{k+1} \neq 0$, and thus $s_{k+1} = t_{k+1} = \omega(\boldsymbol{a}) = \omega(\boldsymbol{a}')$. In this case, $\omega(\boldsymbol{a})$ either is odd and so $s_{k+2} = t_{k+2} = 0$, or is even and so $s_{k+2} = t_{k+2} = \omega(\boldsymbol{a}) + 1$.
In both cases, $s_{k+2} = t_{k+2}$.

□

It is well known that the restriction of $\mathcal{G}_n(m)$ defined in relation (1) to any product space remains a 1-Gray code, see for example [14]. In particular, for $SE_n = \{0\} \times \{0,1\} \times \ldots \times \{0,1,\dots,n-1\}$ we have the next proposition. Its proof is simply based on Lemma 1, Proposition 1, and on the additional remark: for any $\boldsymbol{a} \in SE_k$, $k < n$, it follows that $\omega_{SE}(\boldsymbol{a}) = k$.

Proposition 2. *The list $\mathcal{SE}_n$ is 1-Gray code.*

It is worth to mention that for any statistic st satisfying relations (2) and (3), the list in $\prec$ order for the set of st-restricted growth sequences of length n is an at most 3-Gray code.
Actually, the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, and $\mathcal{S}_n$ are *circular* Gray codes, that is, the last and the first sequences in the list differ in the same way. Indeed, by the definition of $\prec$ order, it follows that:

- $\mathrm{first}(\mathcal{X}_n) = 000\dots 0$;
- $\mathrm{last}(\mathcal{X}_n) = 010\dots 0$;

where $\mathcal{X}_n$ is one of the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, or $\mathcal{S}_n$.

3.2 Generating algorithms for the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, and $\mathcal{S}_n$

Procedure GEN1 in Figure 1 is a general procedure generating exhaustively the list of st-restricted growth sequences, where st is a statistic satisfying relations (2) and (3). According to particular instances of the function OMEGA_X called by it (and so, of the statistic st), GEN1 produces specific st-restricted growth sequences, and in particular the lists $\mathcal{SE}_n$, $\mathcal{A}_n$, $\mathcal{R}_n$, and $\mathcal{S}_n$. From the length one sequence 0, GEN1 constructs recursively increasing length st-restricted growth sequences: for a given prefix $s_1s_2\dots s_k$ it produces all prefixes $s_1s_2\dots s_ki$, with i covering (in increasing or decreasing order) the defining set of $s_1s_2\dots s_k$; and eventually all length n st-restricted growth sequences. It has the following parameters:

- k, the position in the sequence $\boldsymbol{s}$ which is updated by the current call;
- x, belongs to the defining set of $s_1s_2\dots s_{k-1}$, and is the value to be assigned to s_k;
- dir, the direction (ascending for $dir \bmod 2 = 0$ and descending for $dir \bmod 2 = 1$) to cover the defining set of $s_1s_2\dots s_{k-1}$;
- v, the value of the statistic of the prefix $s_1s_2\dots s_{k-1}$ from which the value of the statistic of the current prefix $s_1s_2\dots s_k$ is computed. Remark that $v = \omega_X(s_1s_2\dots s_{k-1}) - 1$.

Function OMEGA_X computes $\omega_X(s_1s_2\dots s_k)$ (see relation (5)), and the main call is GEN1(1,0,0,0).

```
procedure GEN1(k, x, dir, v)
   global: n, s;
   s_k := x;
   if k = n then PRINT s;
   else u := OMEGA_X(v, k);
        if dir mod 2 = 0
        then for i := 0 to u do
               GEN1(k + 1, i, i, u − 1);
        else for i := u to 0 do
               GEN1(k + 1, i, i + 1, u − 1);
end.
```

(a)

```
(i) function OMEGA_SE(w, p)
       return p;
    end.

(ii) function OMEGA_A(w, p)
       if p > 1 and s_p > s_{p−1}
       then return w + 2;
       else return w + 1;
     end.

(iii) function OMEGA_R(w, p)
        if p > 1 and s_p > w
        then return w + 2;
        else return w + 1;
      end.

(iv) function OMEGA_S(w, p)
        return s_p + 1;
     end.
```

(b)

Figure 1: (a) Algorithm GEN1, generating the list $\mathcal{X}_n$; (b) Particular function OMEGA_X called by GEN1, and returning the value for $\omega_X(s_1s_2\dots s_k)$, if X_n is one of the sets: (i) SE_n, (ii) A_n, (iii) R_n, and (iv) S_n.

In GEN1 the amount of computation of each call is proportional with the degree of the call, and there are no degree one calls, and so it satisfies the CAT principle stated at the end of Section 2, and so it is an efficient generating algorithm. The computational tree of GEN1 producing the list $\mathcal{A}_4$ is given in Figure 2. Each node at level k, $1 \le k \le 4$, represents prefixes $s_1 s_2 \dots s_k$, and leaves sequences in $\mathcal{A}_4$.

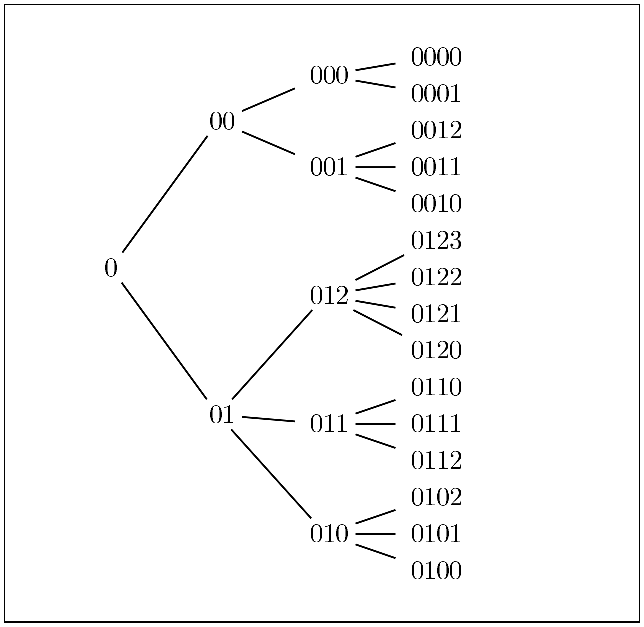

Figure 2: The tree induced by the initial call `Gen1`$(1, 0, 0, 0)$ for $n = 4$ and generating the list $\mathcal{A}_4$.

4 The Co-Reflected Gray Code order for the sets SE_n, A_n, R_n, and S_n

In this section we will consider, as in the previous one, the sets SE_n, A_n, R_n and S_n, but listed in $\prec_c$ order. Our main goal is to prove that the obtained lists are Gray codes as well, and to develop generating algorithms for these lists.

Recall that X_n generically denotes one of the sets SE_n, A_n, R_n, or S_n; and let $\widetilde{\mathcal{X}}_n$ denote their corresponding list in $\prec_c$ order, that are, $\widetilde{\mathcal{SE}}_n$, $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, or $\widetilde{\mathcal{S}}_n$. Clearly, a set of sequences listed in $\prec_c$ order is a *suffix partitioned* list, that is, all sequences with same suffix are contiguous, and such are the lists we consider here.

For a set X_n and a sequence $\boldsymbol{b} = b_{k+1} b_{k+2} \dots b_n$, we call $\boldsymbol{b}$ an *admissible suffix* in X_n if there exists at least a sequence in X_n having suffix $\boldsymbol{b}$. For example, 124 is an admissible suffix in A_6, because there are sequences in A_6 ending with 124, namely 012124 and 010124. On the other hand, 224 is not an admissible suffix in A_6; indeed, there is no length 6 ascent sequence ending with 224.

We denote by $\widetilde{\mathcal{X}}_n \,|\, \boldsymbol{b}$ the sublist of $\widetilde{\mathcal{X}}_n$ of all sequences having suffix $\boldsymbol{b}$, and clearly, $\widetilde{\mathcal{X}}_n \,|\, \boldsymbol{b}$ is a contiguous sublist of $\widetilde{\mathcal{X}}_n$. The set of all x such that $x\boldsymbol{b}$ is also an admissible suffix in X_n is called the *defining set of the suffix* $\boldsymbol{b}$.

For $\prec$ order discussed in Section 3, the characterization of prefixes is straightforward: $a_1 a_2 \dots a_k$ is the prefix of some sequences in X_n, $n > k$, if and only if $a_1 a_2 \dots a_k$ is in X_k.

And the defining set of the prefix $a_1a_2\dots a_k$ is $\{0,1,\dots,\mathsf{st}(a_1a_2\dots a_k)+1\}$. In the case of $\prec_c$ order, it turns out that similar notions are more complicated: for example, 13 is an admissible suffix in A_5, but 13 is not in A_2; and the defining set of the suffix 13 is $\{0,2\}$, because 013 and 213 are both admissible suffixes in A_5, but 113 is not. See Table 4 for the set A_5 listed in $\prec_c$ order.

4.1 Suffix expansion of sequences in the sets SE_n, A_n, R_n, and S_n

For a suffix partitioned list, we need to build st-restricted growth sequences under consideration from right to left, i.e., by expanding their suffix. For this purpose, we need the notions defined below.

Definition 4. Let $\boldsymbol{b}=b_{k+1}b_{k+2}\dots b_n$, $1\le k<n$, an admissible suffix in X_n.

- $\alpha_X(\boldsymbol{b})$ is the set of all elements in the defining set of the suffix $\boldsymbol{b}$. Formally:
$$\alpha_X(\boldsymbol{b})=\{x\,|\,x\boldsymbol{b}\text{ is an admissible suffix in }X_n\},$$
and for the empty suffix ϵ, $\alpha_X(\epsilon)=\{0,1,\dots,n-1\}$.
- $\mu_X(\boldsymbol{b})$ is the minimum required value of the statistic defining the set X_n, and provided by a length $(k+1)$ prefix of a sequence in X_n having suffix $\boldsymbol{b}$. Formally:
$$\mu_X(\boldsymbol{b})=\min\{\mathsf{st}(s_1s_2\dots s_kb_{k+1})\,|\,s_1s_2\dots s_k\boldsymbol{b}\in X_n\}.$$

Notice that $\mu_X(x\boldsymbol{b})\in\{\mu_X(\boldsymbol{b})-1,\mu_X(\boldsymbol{b}),x\}$ for $x\in\alpha_X(\boldsymbol{b})$.

Remark 3. Let st be one of statistics asc, m or lv, and $\boldsymbol{s}=s_1s_2\dots s_n$ be an st-restricted growth sequence. If there is a $k<n$ such that $s_{k+1}=k$, then $s_i=i-1$ for all i, $1\le i\le k$.

Proof. If $s_k<k-1$, then in each case for st, $\mathsf{st}(s_1s_2\dots s_k)<k-1$, which is in contradiction with $s_{k+1}=k$, and so $s_k=k-1$. Similarly, $s_{k-1}=k-2$, …, $s_2=1$, and $s_1=0$. □

Under the conditions in the previous remark, $s_{k+1}=k$ imposes that all values at the left of $k+1$ in $\boldsymbol{s}$ are uniquely determined. As we will see later, in the induced tree of the generating algorithm, all descendants of a node with $s_{k+1}=k$ have degree one, and we will eliminate the obtained degree-one path in order not to alter the algorithm efficiency.

It is routine to check the following propositions. (Actually, Proposition 3 is a consequence of Remark 1.)

Proposition 3. *Let X_n be one of the sets SE_n, A_n, R_n, or S_n. If $\boldsymbol{b}=b_{k+1}b_{k+2}\dots b_n$, $1\le k<n$, is an admissible suffix in X_n, then $b_{k+1}\le\mu_X(\boldsymbol{b})$.*

Proposition 4. *Let Y_n be one of the sets A_n, R_n, or S_n. If $\boldsymbol{b}=b_{k+1}b_{k+2}\dots b_n$, $1\le k<n$, is an admissible suffix in Y_n, then*

1 if $\boldsymbol{b}=b_n$, that is, a length one admissible suffix, then $\mu_Y(\boldsymbol{b})=b_n$;

2 $\mu_Y(\boldsymbol{b})=k$ if and only if $b_{k+1}=k$;

3 if $x\boldsymbol{b}$ is also an admissible suffix in Y_n (i.e., $x\in\alpha_Y(\boldsymbol{b})$) and $x\ge b_{k+1}$, then
$$\mu_Y(x\boldsymbol{b})=\max\{x,\mu_Y(\boldsymbol{b})\}.$$

The following propositions give the values for $\alpha_X(\boldsymbol{b})$ and $\mu_X(x\boldsymbol{b})$, if X_n is one of the sets SE_n, A_n, R_n, or S_n. We do not provide the proofs for Propositions 5, 6, 11, and 12, because they are obviously based on the definition of the corresponding sequences.

Proposition 5. *Let $\boldsymbol{b} = \epsilon$ or $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in SE_n. Then*

$$\alpha_{SE}(\boldsymbol{b}) = \begin{cases} \{0, 1, , \dots, n-1\} & \text{if } \boldsymbol{b} = \epsilon, \\ \{0, 1, \dots, k-1\} & \text{otherwise.} \end{cases}$$

Proposition 6. *Let $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in SE_n and $x \in \alpha_{SE}(\boldsymbol{b})$. Then*

$$\mu_{SE}(x\boldsymbol{b}) = \mu_{SE}(\boldsymbol{b}) - 1.$$

Obviously, for a length one suffix $\boldsymbol{b} = b_n$, it follows that $\mu_{SE}(\boldsymbol{b}) = n - 1$.

Example 2. *If $\boldsymbol{b} = \epsilon$, and $n = 10$, then $\alpha_{SE}(\boldsymbol{b}) = \{0, 1, \dots, 9\}$;
and for $\boldsymbol{b} = b_{10}$, $b_{10} \in \{0, 1, \dots, 9\}$, it follows that $\mu_{SE}(x\boldsymbol{b}) = 9 - 1 = 8$, for all $x \in \alpha_{SE}(\boldsymbol{b})$.*

Proposition 7. *Let $\boldsymbol{b} = \epsilon$ or $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in A_n. Then*

$$\alpha_A(\boldsymbol{b}) = \begin{cases} \{0, 1, \dots, n-1\} & \text{if} \quad \boldsymbol{b} = \epsilon, \\ \{k-1\} & \text{if} \quad \mu_A(\boldsymbol{b}) = k, \text{ or } \mu_A(\boldsymbol{b}) = k-1 \text{ and } b_{k+1} = 0, \\ \{0, 1, \dots, b_{k+1} - 1\} \cup \{k-1\} & \text{if} \quad \mu_A(\boldsymbol{b}) = k-1 \text{ and } 0 < b_{k+1} < k, \\ \{0, 1, \dots, k-1\} & \text{if} \quad \mu_A(\boldsymbol{b}) < k-1. \end{cases}$$

Proof. If $\boldsymbol{b} = \epsilon$, the result is obvious.
For $\boldsymbol{b} \neq \epsilon$, let $x \in \alpha_A(\boldsymbol{b})$.
If $\mu_A(\boldsymbol{b}) = k$, by Proposition 4 point 2, $b_{k+1} = k$ and by Remark 3 we have $x = k - 1$.
If $\mu_A(\boldsymbol{b}) = k - 1$ and $b_{k+1} = 0$, then $\mathsf{asc}(xb_{k+1}) = 0$, and so $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) = k - 1$, and again by Proposition 4 point 2 we have $x = k - 1$.
If $\mu_A(\boldsymbol{b}) = k - 1$ and $0 < b_{k+1} < k$, then there are two possibilities for $\mu_A(x\boldsymbol{b})$:

- $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) = k - 1$, if $\mathsf{asc}(xb_{k+1}) = 0$, and as above $x = k - 1$;
- $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) - 1 = k - 2$, if $\mathsf{asc}(xb_{k+1}) = 1$. In this case $x \in \{0, 1, \dots, b_{k+1} - 1\}$.

If $\mu_A(\boldsymbol{b}) < k - 1$ (and consequently $0 \leq b_{k+1} < k$), then there are two possibilities for $\mu_A(x\boldsymbol{b})$:

- $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b})$, if $\mathsf{asc}(xb_{k+1}) = 0$, and we have $x \in \{b_{k+1}, b_{k+1} + 1, \dots, k - 1\}$;
- $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) - 1$, if $\mathsf{asc}(xb_{k+1}) = 1$, and we have $x \in \{0, 1, \dots, b_{k+1} - 1\}$.

□

Proposition 8. *Let $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in A_n and $x \in \alpha_A(\boldsymbol{b})$. Then*

$$\mu_A(x\boldsymbol{b}) = \begin{cases} x & \text{if} \quad x \geq \mu_A(\boldsymbol{b}), \\ \mu_A(\boldsymbol{b}) & \text{if} \quad b_{k+1} \leq x < \mu_A(\boldsymbol{b}), \\ \mu_A(\boldsymbol{b}) - 1 & \text{if} \quad x < b_{k+1}. \end{cases}$$

Proof. If $x \geq \mu_A(\boldsymbol{b})$, by Proposition 3 it follows that $x \geq b_{k+1}$, and by Proposition 4 point 3, that $\mu_A(x\boldsymbol{b}) = \max\{x, \mu_A(\boldsymbol{b})\} = x$.
If $b_{k+1} \leq x < \mu_A(\boldsymbol{b})$, then, again by Proposition 4 point 3, it follows that $\mu_A(x\boldsymbol{b}) = \max\{x, \mu_A(\boldsymbol{b})\} = \mu_A(\boldsymbol{b})$.
If $x < b_{k+1}$, then $\mathsf{asc}(xb_{k+1}) = 1$, so $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) - 1$. □

Example 3. *Let $k = 5$, $n = 9$, and $\boldsymbol{b} = b_6b_7b_8b_9 = 2050$ be an admissible suffix in A_9. Clearly, $\mu_A(\boldsymbol{b})$, the minimum number of ascents in a prefix $s_1s_2\dots s_5b_6$ such that $s_1s_2\dots s_5\boldsymbol{b} \in A_9$, is 4. In this case, denoting s_5 by x, we have*

- *the set $\alpha_A(\boldsymbol{b})$ of all possible values for x is $\{0, 1, , \dots, b_{k+1} - 1\} \cup \{k - 1\} = \{0, 1\} \cup \{4\}$.*
- *$\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) - 1 = 4 - 1 = 3$, if $x \in \{0, 1\}$; or $\mu_A(x\boldsymbol{b}) = \mu_A(\boldsymbol{b}) = 4$, if $x = 4$.*

Proposition 9. *Let $\boldsymbol{b} = \epsilon$ or $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in R_n. Then*

$$\alpha_R(\boldsymbol{b}) = \begin{cases} \{0, 1, \dots, n-1\} & \text{if} \quad \boldsymbol{b} = \epsilon, \\ \{k-1\} & \text{if} \quad \mu_R(\boldsymbol{b}) = k, \text{ or } \mu_R(\boldsymbol{b}) = k-1 \text{ and } b_{k+1} < k-1, \\ \{0, 1, \dots, k-1\} & \text{if} \quad \mu_R(\boldsymbol{b}) = k-1 \text{ and } b_{k+1} = k-1, \text{ or } \mu_R(\boldsymbol{b}) < k-1. \end{cases}$$

Proof. If $\boldsymbol{b} = \epsilon$, the result is obvious.
For $\boldsymbol{b} \neq \epsilon$, let $x \in \alpha_R(\boldsymbol{b})$.
If $\mu_R(\boldsymbol{b}) = k$, by Proposition 4 point 2, $b_{k+1} = k$ and by Remark 3 we have $x = k - 1$.
If $\mu_R(\boldsymbol{b}) = k - 1$ and $b_{k+1} < k - 1$, then the maximal value of the statistic m (defining the set R_n) of a length $k + 1$ prefix ending with $b_{k+1} < k - 1$ is $k - 1$, and it is reached when $x = k - 1$.
If $\mu_R(\boldsymbol{b}) = k - 1$ and $b_{k+1} = k - 1$, then there are two possibilities for $\mu_R(x\boldsymbol{b})$:

- $\mu_R(x\boldsymbol{b}) = \mu_R(\boldsymbol{b}) = k - 1$, and as above, this implies $x = k - 1$;
- $\mu_R(x\boldsymbol{b}) = \mu_R(\boldsymbol{b}) - 1 = k - 2$, which implies $x \in \{0, 1, \dots, k - 2\}$.

Finally, if $\mu_R(\boldsymbol{b}) < k - 1$, then x can be any value in $\{0, 1, \dots, k - 1\}$. □

Proposition 10. *Let $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in R_n and $x \in \alpha_R(\boldsymbol{b})$. Then*

$$\mu_R(x\boldsymbol{b}) = \begin{cases} x & \text{if} \quad x \geq \mu_R(\boldsymbol{b}), \\ \mu_R(\boldsymbol{b}) & \text{if} \quad b_{k+1} \leq x < \mu_R(\boldsymbol{b}) \text{ or } x < b_{k+1} < \mu_R(\boldsymbol{b}), \\ \mu_R(\boldsymbol{b}) - 1 & \text{if} \quad x < b_{k+1} = \mu_R(\boldsymbol{b}). \end{cases}$$

Proof. The case $x \geq \mu_R(\boldsymbol{b})$ is analogous with the similar case in Proposition 8.
The next case is equivalent with $x < \mu_R(\boldsymbol{b})$ and $b_{k+1} < \mu_R(\boldsymbol{b})$, and since R_n corresponds to the statistic m, the result holds.
Finally, if $x < b_{k+1} = \mu_R(\boldsymbol{b})$, then $\mu_R(\boldsymbol{b}) = \mu_R(x\boldsymbol{b}) + 1$, and so $\mu_R(x\boldsymbol{b}) = \mu_R(\boldsymbol{b}) - 1$. □

Example 4. *Let $k = 4$, $n = 7$, and $\boldsymbol{b} = b_5b_6b_7 = 241$ be an admissible suffix in R_7. It follows that $b_{k+1} = 2$, $\mu_R(\boldsymbol{b}) = 3$ and*

- *$\alpha_R(\boldsymbol{b}) = \{k - 1\} = \{3\}$;*
- *$\mu_R(x\boldsymbol{b}) = x = 3$.*

Proposition 11. *Let $\boldsymbol{b} = \epsilon$ or $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in S_n. Then*

$$\alpha_S(\boldsymbol{b}) = \begin{cases} \{0, 1, \dots, n-1\} & \text{if} \quad \boldsymbol{b} = \epsilon, \\ \{k-1\} & \text{if} \quad b_{k+1} = k, \\ \{C, C+1, \dots, k-1\} & \text{if} \quad 0 \leq b_{k+1} \leq k-1, \end{cases}$$

where $C = \max\{0, b_{k+1} - 1\}$.

Since $\mu_S(\boldsymbol{b}) = b_{k+1}$, the next result follows:

Proposition 12. *Let $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in S_n and $x \in \alpha_S(\boldsymbol{b})$. Then*

$$\mu_S(x\boldsymbol{b}) = x.$$

Example 5. *Let $k = 6$, $n = 9$, and $\boldsymbol{b} = b_7b_8b_9 = 457$ be an admissible suffix in S_9. So we have $\mu_S(\boldsymbol{b}) = 4$, $C = \max\{0, b_{k+1} - 1\} = \max\{0, 3\} = 3$, and*

- $\alpha_S(\boldsymbol{b}) = \{C, C+1, \dots, k-1\} = \{3, 4, 5\}$;
- $\mu_S(x\boldsymbol{b}) = x$, *where* $x \in \{3, 4, 5\}$.

4.2 The bound of Hamming distance between successive sequences in the lists $\widetilde{\mathcal{SE}}_n$, $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, and $\widetilde{\mathcal{S}}_n$

Now we show that the Hamming distance between two successive sequences in the mentioned lists is bounded from above by a constant, which implies that the lists are Gray codes. These results are embodied in Theorems 2, 3 and 4.

4.2.1 The list $\widetilde{\mathcal{SE}}_n$

Theorem 2. *The list $\widetilde{\mathcal{SE}}_n$ is 1-Gray code.*

Proof. The result follows from the fact that the restriction of the 1-Gray code list $\mathcal{G}_n(n)$ to any product space remains a 1-Gray code (see [14]), in particular to the set

$$V_n = \vartheta_1 \times \vartheta_2 \times \dots \times \vartheta_n,$$

where

- $\vartheta_i = \{0, 1, ..., n-i\}$, if n is odd and i is even, or
- $\vartheta_i = \{i-1, i, ..., n-1\}$, if n is even, or n and i are both odd.

Then by applying to each sequence $\boldsymbol{s}$ in the list $\mathcal{V}_n$ the two transforms mentioned before Definition 2, namely:

- complementing each digit in $\boldsymbol{s}$ if n is even, or only digits in odd positions if n is odd, then
- reversing the obtained sequence,

the desired 1-Gray code for the set SE_n in $\prec_c$ order is obtained. ☐

4.2.2 The lists $\widetilde{\mathcal{A}}_n$ and $\widetilde{\mathcal{R}}_n$

The next proposition describes the pattern of $\boldsymbol{s} = \text{last}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$ and $\boldsymbol{s} = \text{first}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$, where $\widetilde{\mathcal{Y}}_n$ is one of the lists $\widetilde{\mathcal{A}}_n$ or $\widetilde{\mathcal{R}}_n$.

Proposition 13. *Let Y_n be one of the sets A_n or R_n, and $\boldsymbol{b} = b_{k+1}b_{k+2}\dots b_n$ be an admissible suffix in Y_n. If $\boldsymbol{s} = \text{last}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$ or $\boldsymbol{s} = \text{first}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$, then $\boldsymbol{s}$ has one of the following patterns:*

- $\boldsymbol{s} = 012\dots(k-2)(k-1)\boldsymbol{b}$, *or*

- $\boldsymbol{s} = 012\dots(k-2)0\boldsymbol{b}$.

Proof. Let $\boldsymbol{s} = s_1 s_2 \dots s_k b_{k+1} b_{k+2} \dots b_n$. Since $\boldsymbol{s} = \mathrm{last}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$ or $s = \mathrm{first}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$, according to $\alpha_Y(\boldsymbol{b})$ given in Propositions 7 and 9, it follows that $s_k \in \{0, k-1\}$. In other words, s_k is either the smallest or the largest value in $\alpha_Y(\boldsymbol{b})$.
If $s_k = k-1$, then by Remark 3 we have $\boldsymbol{s} = 012\dots(k-2)(k-1)\boldsymbol{b}$.
If $s_k = 0$, then considering the definition of $\prec_c$ order we have either

- $\boldsymbol{s} = \mathrm{first}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$ and $\sum_{i=k+1}^{n} b_i + (n-k)$ is odd, or
- $\boldsymbol{s} = \mathrm{last}(\widetilde{\mathcal{Y}}_n \mid \boldsymbol{b})$ and $\sum_{i=k+1}^{n} b_i + (n-k)$ is even.

For the first case, again by the definition of $\prec_c$ order, it follows that s_{k-1} must be the largest value in $\alpha_Y(0\boldsymbol{b})$, and so $s_{k-1} = k-2$, and by Remark 3, $\boldsymbol{s} = 012\dots(k-2)0\boldsymbol{b}$. Similarly, the same result is obtained for the second case. □

A direct consequence of the previous proposition is that $\prec_c$ order gives a more restrictive Gray codes than those given by $\prec$ order for the sets A_n and R_n. This is formalized in the next theorem.

Theorem 3. *The lists $\widetilde{\mathcal{A}}_n$ and $\widetilde{\mathcal{R}}_n$ are 2-adjacent Gray codes.*

Proof. Let $\boldsymbol{s}, \boldsymbol{t} \in Y_n$, with $\boldsymbol{t} = \widetilde{\mathrm{succ}}_Y(\boldsymbol{s})$. If $k+1$ is the rightmost position where $\boldsymbol{s}$ and $\boldsymbol{t}$ differ, then there are admissible suffixes $\boldsymbol{b} = b_{k+1} b_{k+2} \dots b_n$ and $\boldsymbol{b}' = b'_{k+1} b_{k+2} \dots b_n$ in Y_n such that $\boldsymbol{s} = \mathrm{last}(\widetilde{\mathcal{Y}}_n \,|\boldsymbol{b})$ and $\boldsymbol{t} = \mathrm{first}(\widetilde{\mathcal{Y}}_n \,|\boldsymbol{b}')$.
By Proposition 13, $\boldsymbol{s}$ has pattern

$$\begin{array}{l} 012\dots(k-2)(k-1)\boldsymbol{b}, \text{ or} \\ 012\dots(k-2)0\boldsymbol{b}; \end{array}$$

and $\boldsymbol{t}$ has pattern

$$\begin{array}{l} 012\dots(k-2)(k-1)\boldsymbol{b}', \text{ or} \\ 012\dots(k-2)0\boldsymbol{b}'. \end{array}$$

And in any case, $\boldsymbol{s}$ and $\boldsymbol{t}$ differ in position $k+1$ and possibly in position k.

□

4.2.3 The list $\widetilde{\mathcal{S}}_n$

The next proposition gives the pattern of $\mathrm{last}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$ and $\mathrm{first}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$ for an admissible suffix $\boldsymbol{b}$ in S_n.

Proposition 14. *Let $\boldsymbol{b} = b_{k+1} b_{k+2} \dots b_n$ be an admissible suffix in S_n. If $\boldsymbol{s} = \mathrm{last}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$, then the pattern of $\boldsymbol{s}$ is given by:*

- *if $b_{k+1} = k$ or $\sum_{i=k+1}^{n} b_i + (n-k)$ is odd, then*

$$\boldsymbol{s} = 012\dots(k-2)(k-1)\boldsymbol{b};$$

- *if $b_{k+1} < k$ and $\sum_{i=k+1}^{n} b_i + (n-k)$ is even, and either $b_{k+1} = 0$ or b_{k+1} is odd, then*

$$\boldsymbol{s} = 012\dots(k-2)(\max\{0, b_{k+1}-1\})\boldsymbol{b};$$

- *if* $b_{k+1} < k$ *and* $\sum_{i=k+1}^{n} b_i + (n-k)$ *is even, and* $b_{k+1} > 0$ *is even, then*

$$\boldsymbol{s} = 012\ldots(k-3)(b_{k+1}-2)(b_{k+1}-1)\boldsymbol{b}.$$

Similar results hold for $\boldsymbol{s} = \text{first}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$ *by replacing 'odd' by 'even', and* vice versa, *for the parity of* $\sum_{i=k+1}^{n} b_i + (n-k)$.

Proof. Let $\boldsymbol{s} = s_1 s_2 \ldots s_k b_{k+1} \ldots b_n = \text{last}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$.
If $b_{k+1} = k$ or $\sum_{i=k+1}^{n} b_i + (n-k)$ is odd, then s_k is the largest value in $\alpha_S(\boldsymbol{b})$, so $s_k = k-1$, and by Remark 3, $s_i = i-1$ for $1 \le i \le k$. So the first case holds.
If $\sum_{i=k+1}^{n} b_i + (n-k)$ is even and $b_{k+1} < k$, then s_k is the smallest value in $\alpha_S(\boldsymbol{b})$, namely $\max\{0, b_{k+1}-1\}$, which is even if $b_{k+1} = 0$ or b_{k+1} is odd. Thus, by the definition of $\prec_c$ order, s_{k-1} is the largest value in $\alpha_S(s_k\boldsymbol{b})$, which is $k-2$, and by Remark 3, the second case holds.
For the last case, as above, $s_k = \max\{0, b_{k+1}-1\}$, and considering $b_{k+1} > 0$ and even, it follows that $s_k = b_{k+1}-1$ is odd. Thus s_{k-1} is the minimal value in $\alpha_S(s_k\boldsymbol{b})$, that is $b_{k+1}-2$, which in turns is even, and the last case holds.
The proof for the case $\boldsymbol{s} = \text{first}(\widetilde{\mathcal{S}}_n \mid \boldsymbol{b})$ is similar. ☐

Theorem 4. *The list* $\widetilde{\mathcal{S}}_n$ *is* 3*-adjacent Gray codes.*

Proof. Let $\boldsymbol{t} = \widetilde{\text{succ}}_S(\boldsymbol{s})$, and $k+1$ be the rightmost position where $\boldsymbol{s}$ and $\boldsymbol{t}$ differ. Let us denote by $\boldsymbol{b}$ the length $(n-k)$ suffix of $\boldsymbol{s}$ and $\boldsymbol{b}'$ that of $\boldsymbol{t}$; so, $\boldsymbol{s} = \text{last}(\boldsymbol{b} \mid \mathcal{S}_n)$ and $\boldsymbol{t} = \text{first}(\boldsymbol{b}' \mid \mathcal{S}_n)$. It follows by Proposition 14, that $s_i = t_i = i-1$ for all $i \le k-2$, and so the other differences possibly occur in position k and in position $k-1$.

Considering all valid combinations for $\boldsymbol{s}$ and $\boldsymbol{t}$ as given in Proposition 14, the proof of the adjacency is routine, and based on the following: $s_k \ne t_k$ if and only if $s_{k-1} \ne t_{k-1}$. It follows that $\boldsymbol{s}$ and $\boldsymbol{t}$ differ in one position, or three positions which are adjacent.

☐

In addition, the lists $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, and $\widetilde{\mathcal{S}}_n$, are circular Gray codes. This is a consequence of the following remarks based on Propositions 13 and 14:

- $\text{first}(\widetilde{\mathcal{Y}}_n) = 012\ldots(n-2)(n-1)$;
- $\text{last}(\widetilde{\mathcal{Y}}_n) = 012\ldots(n-2)0$;

where $\widetilde{\mathcal{Y}}_n$ is one of the lists $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, or $\widetilde{\mathcal{S}}_n$.

4.3 Generating algorithm for $\widetilde{\mathcal{SE}}_n$, $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, and $\widetilde{\mathcal{S}}_n$

Here we explain algorithm Gen2 in Figure 3 which generates suffix partitioned Gray codes for restricted growth sequences; according to particular instances of the functions called by it, Gen2 produces the list $\widetilde{\mathcal{SE}}_n$, $\widetilde{\mathcal{A}}_n$, $\widetilde{\mathcal{R}}_n$, or $\widetilde{\mathcal{S}}_n$. Actually, for convenience, Gen2 produces length $(n+1)$ sequences $\boldsymbol{s} = s_1 s_2 \ldots s_{n+1}$ with $s_{n+1} = 0$, and so, neglecting the last value in each sequence $\boldsymbol{s}$ the desired list is obtained. Notice that with this dummy value for s_{n+1} we have $\mu_X(s_k s_{k+1} \ldots s_n) = \mu_X(s_k s_{k+1} \ldots s_n 0)$, for $k \le n$, and similarly for α_X.

In Gen2, the sequence $\boldsymbol{s}$ is a global variable, and initialized by $01\ldots(n-1)0$, which is the first length n sequence in $\prec_c$ order, followed by a 0; and the main call is Gen2$(n+1, 0, 0, 0)$. Procedure Gen2 has the following parameters (the first three of them are similar with those of procedure Gen1):

- k, the position in the sequence $\boldsymbol{s}$ which is updated by the current call;
- x, the value to be assigned to s_k;
- dir, gives the direction in which s_{k-1} covers $\alpha_X(s_k s_{k+1} \dots s_n 0)$, the defining set of the current suffix;
- v, the value of $\mu_X(s_{k+1} \dots s_n 0)$.

The functions called by Gen2 are given in Figures 4 and 5. They are principally based on the evaluation of α_X and μ_X for the current suffix of $\boldsymbol{s}$, and are:

- Mu_$X(k, x, v)$ returns the value of $\mu_X(s_k s_{k+1} \dots s_n 0)$, with $x = s_k$.
- IsDegreeOne_$X(k)$ stops the recursive calls when $\alpha_X(s_k s_{k+1} \dots s_n 0)$ has only one element, namely $k-2$. In this case, by Remark 3 the sequence is uniquely determined by the current suffix, and this prevents Gen2 to produce degree one calls. In addition, IsDegreeOne_$X(k)$ sets appropriately $d-1$ values at the left of s_k, where d is the upper bound of the Hamming distance in the list (changes at the left of s_1 are considered with no effect). This can be considered as a *Path Elimination Technique* or PET (see [16]).
- Lowest_$X(k)$ is called when IsDegreeOne_X returns false, and gives the lowest value in $\alpha_X(s_k s_{k+1} \dots s_n 0)$.
- SecLargest_$X(k, u)$, is called when IsDegreeOne_X returns false, and gives the second largest value in $\alpha_X(s_k s_{k+1} \dots s_n 0)$ (the largest value being always $k - 2$).

By this construction, algorithm Gen2 has no degree one calls and it satisfies the CAT principle. Figure 6 shows the tree induced by the algorithm when generates $\widetilde{\mathcal{A}}_4$.

```
procedure Gen2(k, x, dir, v)
   global n,s;
   s_k := x;
   u :=Mu_X(k, x, v);
   if IsDegreeOne_X(k, u) then Print s;
   else c:=Lowest_X(k);
          d:=SecLargest_X(k, u);
          if dir mod 2 = 1 then
             for i := c to d do
                  Gen2(k − 1, i, i, u);
          Gen2(k − 1, k − 2, k − 1 − (dir mod 2), u);
          if dir mod 2 = 0 then
             for i := d downto c do
                  Gen2(k − 1, i, i + 1, u);
end.
```

Figure 3: Algorithm Gen2, generating the list $\widetilde{\mathcal{X}}_n$.

```
function Mu_A(m, i, w)
   if i ≥ w then return i;
   else if i ≥ s_{m+1}
         then return w;
         else return w − 1;
end.

function IsDegreeOne_A(m, v)
   if v = m − 1 or
      (v = m − 2 and s_m = 0)
   then s_{m−1} := m − 2;
         return true;
   else return false;
end.

function Lowest_A(m)
   return 0;
end.

function SecLargest_A(m, w)
   if w = m − 2 and s_m > 0
      and s_m < m − 1
   then return s_m − 1;
   else return m − 3;
end.
```

(a)

```
function Mu_R(m, i, w)
   if i ≥ w then return i;
   else if s_{m+1} < w
         then return w;
         else return w − 1;
end.

function IsDegreeOne_R(m, v)
   if v = m − 1 or
      (v = m − 2 and s_m < m − 2)
   then s_{m−1} := m − 2;
         return true;
   else return false;
end.

function Lowest_R(m)
   return 0;
end.

function SecLargest_R(m, w)
   return m − 3;
end.
```

(b)

Figure 4: Particular functions called by Gen2, generating the lists: (a) $\widetilde{\mathcal{A}}_n$, and (b) $\widetilde{\mathcal{R}}_n$.

```
function Mu_S(m, i, w)
   return i;
end.

function IsDegreeOne_S(m, v)
   if s_m = m − 1
   then s_{m−1} := m − 2;
        s_{m−2} := m − 3;
        return true;
   else return false;
end.

function Lowest_S(m)
   if s_m > 1 and s_m ≤ m − 2
   then return s_m − 1;
   else return 0;
end.

function SecLargest_S(m, w)
   return m − 3;
end.
```

(a)

```
function Mu_SE(m, i, w)
   if m = n then return n − 1;
   else return w − 1;
end.

function IsDegreeOne_SE(m, v)
   if m = 2 then return true;
   else return false;
end.

function Lowest_SE(m)
   return 0;
end.

function SecLargest_SE(m, w)
   return m − 3;
end.
```

(b)

Figure 5: Particular functions called by Gen2, generating the lists: (a) $\widetilde{\mathcal{S}}_n$, and (b) $\widetilde{\mathcal{SE}}_n$.

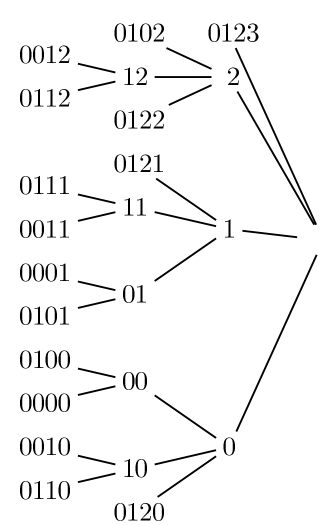

Figure 6: The tree induced by the call `Gen2`$(5, 0, 0, 0)$, generating the list $\widetilde{\mathcal{A}}_4$.

5 Final remarks

We conclude this paper by comparing for each of the sets SE_n, A_n, R_n and S_n the prefix partitioned Gray codes induced by $\prec$ order and the suffix partitioned one induced by $\prec_c$ order. This can be done by comparing the Hamming distance between all pairs of successive sequences either in the worst case or in average.

Table 1 summarizes Theorems 1, 3, and 4, and Proposition 2, and gives the upper bound of the Hamming distance (that is, the worst case Hamming distance) for the two order relations. It shows that for the sets SE_n and S_n these relations have same performances, and for the sets A_n and R_n, $\prec_c$ order induces more restrictive Gray codes.

Set	$\prec$ (RGC) order	$\prec_c$ (Co-RGC) order
SE_n	1	1
A_n	3	2
R_n	3	2
S_n	3	3

Table 1: The bound of the Hamming distance between two successive sequences in $\prec$ and $\prec_c$ orders.

For a list $\mathcal{L}$ of sequences, the *average Hamming distance* is defined as

$$\frac{\sum d(\boldsymbol{s},\boldsymbol{t})}{N-1},$$

where the summation is taken over all $\boldsymbol{s}$ in $\mathcal{L}$, except its last element, $\boldsymbol{t}$ is the successor of $\boldsymbol{s}$ in $\mathcal{L}$, d is the Hamming distance, and N the number of sequences in $\mathcal{L}$.

Surprisingly, despite $\prec_c$ order has same or better performances in terms of worst case Hamming distance, if we consider the average Hamming distance, numerical evidences show that $\prec$ order is 'more optimal' than $\prec_c$ order on A_n, R_n ($n \geq 5$), and S_n ($n \geq 6$). And this phenomenon strengthens for large n; see Table 2.

n	$\prec$ (RGC) order				$\prec_c$ (Co-RGC) order			
	$\mathcal{SE}_n$	$\mathcal{A}_n$	$\mathcal{R}_n$	$\mathcal{S}_n$	$\widetilde{\mathcal{SE}}_n$	$\widetilde{\mathcal{A}}_n$	$\widetilde{\mathcal{R}}_n$	$\widetilde{\mathcal{S}}_n$
4	1	1.21	1.21	1.31	1	1.14	1.14	1.15
5	1	1.13	1.12	1.29	1	1.19	1.18	1.24
6	1	1.09	1.07	1.27	1	1.23	1.20	1.31
7	1	1.06	1.06	1.26	1	1.25	1.22	1.35
8	1	1.04	1.04	1.25	1	1.26	1.23	1.37
9	1	1.03	1.03	1.24	1	1.28	1.24	1.39
10	1	1.02	1.03	1.23	1	1.28	1.24	1.41

Table 2: The average Hamming distance for $\prec$ order and $\prec_c$ order.

Algorithmically, $\prec_c$ order has the advantage that its corresponding generating algorithm, Gen2, is more appropriate to be parallelized than its $\prec$ order counterpart, Gen1. Indeed, the main call of Gen2 produces n recursive calls (compare to two recursive calls produced by the main call of Gen1), and so we can have more parallelized computations; and this is more suitable for large n. See Figure 2 and 6 for examples of computational trees.

Finally, it will be of interest to explore order relation based Gray codes for restricted growth sequences defined by statistics other than those considered in this paper. In this vein we suggest the following conjecture, checked by computer for $n \leq 10$, and concerning descent sequences (defined similarly with ascent sequences in Section 2).

Conjecture 1. *The set of length n descent sequences listed in $\prec_c$ order is a 4-adjacent Gray code.*

Appendix

Sequence	S_5	R_5	A_5
00000	✓	✓	✓
00001	✓	✓	✓
00012	✓	✓	✓
00011	✓	✓	✓
00010	✓	✓	✓
00123	✓	✓	✓
00122	✓	✓	✓
00121	✓	✓	✓
00120	✓	✓	✓
00110	✓	✓	✓
00111	✓	✓	✓
00112	✓	✓	✓
00102		✓	✓
00101	✓	✓	✓
00100	✓	✓	✓
01230	✓	✓	✓
01231	✓	✓	✓
01232	✓	✓	✓

Sequence	S_5	R_5	A_5
01233	✓	✓	✓
01234	✓	✓	✓
01223	✓	✓	✓
01222	✓	✓	✓
01221	✓	✓	✓
01220	✓	✓	✓
01210	✓	✓	✓
01211	✓	✓	✓
01212	✓	✓	✓
01213		✓	✓
01203		✓	✓
01202		✓	✓
01201	✓	✓	✓
01200	✓	✓	✓
01100	✓	✓	✓
01101	✓	✓	✓
01102		✓	✓
01112	✓	✓	✓

Sequence	S_5	R_5	A_5
01111	✓	✓	✓
01110	✓	✓	✓
01120	✓	✓	✓
01121	✓	✓	✓
01122	✓	✓	✓
01123	✓	✓	✓
01023		✓	✓
01022		✓	✓
01021		✓	✓
01020		✓	✓
01010	✓	✓	✓
01011	✓	✓	✓
01012	✓	✓	✓
01013			✓
01002		✓	✓
01001	✓	✓	✓
01000	✓	✓	✓

Table 3: The sets S_5, R_5, and A_5 listed in $\prec$ order.

Sequence	S_5	R_5	A_5
01234	✓	✓	✓
01233	✓	✓	✓
01023		✓	✓
00123	✓	✓	✓
01123	✓	✓	✓
01223	✓	✓	✓
01213		✓	✓
01013			✓
01203		✓	✓
01202		✓	✓
01102		✓	✓
00102		✓	✓
01002		✓	✓
01012	✓	✓	✓
00012	✓	✓	✓
00112	✓	✓	✓
01112	✓	✓	✓
01212	✓	✓	✓

Sequence	S_5	R_5	A_5
01222	✓	✓	✓
01122	✓	✓	✓
00122	✓	✓	✓
01022		✓	✓
01232	✓	✓	✓
01231	✓	✓	✓
01021		✓	✓
00121	✓	✓	✓
01121	✓	✓	✓
01221	✓	✓	✓
01211	✓	✓	✓
01111	✓	✓	✓
00111	✓	✓	✓
00011	✓	✓	✓
01011	✓	✓	✓
01001	✓	✓	✓
00001	✓	✓	✓
00101	✓	✓	✓

Sequence	S_5	R_5	A_5
01101	✓	✓	✓
01201	✓	✓	✓
01200	✓	✓	✓
01100	✓	✓	✓
00100	✓	✓	✓
00000	✓	✓	✓
01000	✓	✓	✓
01010	✓	✓	✓
00010	✓	✓	✓
00110	✓	✓	✓
01110	✓	✓	✓
01210	✓	✓	✓
01220	✓	✓	✓
01120	✓	✓	✓
00120	✓	✓	✓
01020		✓	✓
01230	✓	✓	✓

Table 4: The sets S_5, R_5, and A_5 listed in $\prec_c$ order.

References

[1] Gray, F. (1953) *Pulse Code Communication*. U.S. Patent 2632058.

[2] Er, M.C. (1984) On generating the N-ary reflected Gray code. *IEEE Transactions on Computers*, **33**, 739–741.

[3] Li, Y., Sawada, J. (2009) Gray codes for reflectable languages. *Information Processing Letters*, **5**, 296–300.

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

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

[6] Ruskey, F. and Williams, A. (2009) The coolest way to generate combinations. *Discrete Mathematics*, **309**, 5305–5320.

[7] Ruskey, F., Sawada, J. and Williams, A. (2012) Binary bubble languages. *Journal of Combinatorial Theory Series A,* **119**, 155–169.

[8] Klingsberg, P. (1981) A Gray code for compositions. *Journal of Algorithms*, **3**, 41–44.

[9] Walsh, T. (2000) Loop-free sequencing of bounded integer compositions. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **33**, 323–345.

[10] Vajnovszki, V. (2001) A loopless generation of bitstrings without p consecutive ones. *Proceedings of the Third International Conference on Combinatorics, Computability and Logic, (DMTCS '01)*, Constanta, Romania, 2-6 July, pp. 227–240. Springer-Verlag, London.

[11] Baril, J.-L. and Vajnovszki, V. (2005) Minimal change list for Lucas strings and some graph theoretic consequences. *Theoretical Computer Science*, **346**, 189–199.

[12] Vajnovszki, V. (2007) Gray code order for Lyndon words. *Discrete Mathematics and Theoretical Computer Science*, **9**, 145–152.

[13] Vajnovszki, V. (2008) More restrictive Gray codes for necklaces and Lyndon words. *Information Processing Letters*, **106**, 96–99.

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

[15] Ruskey, F., Savage, C. (1994) Gray codes for set partitions and restricted growth tails. *Australasian Journal of Combinatorics*, **10**, 85–96.

[16] Ruskey, F. *Combinatorial Generation*. Book in preparation.

[17] Mansour, T. and Vajnovszki, V. (2013) Efficient generation of restricted growth words. *Information Processing Letters*, **113**, 613–616.

[18] Bousquet-Mélou, M., Claesson, A., Dukes, M. and Kitaev, S. (2010) (2 + 2)-free posets, ascent sequences and pattern avoiding permutations. *Journal of Combinatorial Theory Series A*, **7**, 884–909.

[19] Dukes, M. and Parviainen R. (2010) Ascent Sequences and Upper Triangular Matrices Containing Non-Negative Integers. *Electronic Journal of Combinatorics*, **17**, R53.

[20] Duncan, P. and Steingrímsson E. (2011) Pattern Avoidance in Ascent Sequences. *Electronic Journal of Combinatorics*, **18**, P226.

[21] Chen, W.Y.V., Dai, A.Y.L., Dokos, T., Dwyer, T. and Sagan, B.E. (2013) On 021-Avoiding Ascent Sequences. *Electronic Journal of Combinatorics*, **20**, P76.

[22] Mansour, M. and Shattuck, M. (2014) Some enumerative results related to ascent sequences. *Discrete Mathematics*, **315-316**, 29–41.

[23] Kitaev, S. (2011) *Patterns in Permutations and Words*, Springer-Verlag, Berlin.

[24] Stanley, R.P. (1999) *Enumerative Combinatorics Vol. 2*. Cambridge University Press, New York.

[25] van Baronaigien, D.R. and Ruskey, F. (1993) Efficient generation of subsets with a given sum. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **14**, 87–96.