# Counting and generating permutations using timed languages<sup>\*</sup> \*\*

Nicolas Basset

Department of computer science, University of Oxford, United Kingdom basset@cs.ox.ac.uk

Abstract. The signature of a permutation  $\sigma$  is a word  $\mathbf{sg}(\sigma) \subseteq \{\mathbf{a}, \mathbf{d}\}^*$ whose  $i^{th}$  letter is  $\mathbf{d}$  when  $\sigma$  has a descent (i.e.  $\sigma(i) > \sigma(i+1)$ ) and is  $\mathbf{a}$  when  $\sigma$  has an ascent (i.e.  $\sigma(i) < \sigma(i+1)$ ). Combinatorics of permutations with a prescribed signature is quite well explored. Here we state and address the two problems of counting and randomly generating in the set  $\mathbf{sg}^{-1}(L)$  of permutations with signature in a given regular language  $L \subseteq \{\mathbf{a}, \mathbf{d}\}^*$ . First we give an algorithm that computes a closed form formula for the exponential generating function of  $\mathbf{sg}^{-1}(L)$ . Then we give an algorithm that generates randomly the *n*-length permutations of  $\mathbf{sg}^{-1}(L)$  in a uniform manner, that is all the permutations of a given length with signature in L are equally probable to be returned. Both contributions are based on a geometric interpretation of a subclass of regular timed languages.

Generating all the permutations with a prescribed signature (described in the abstract) or simply counting them are two classical combinatorial topics (see [17] and reference therein). The random generation of permutations with a prescribed signature has been addressed very recently by Philippe Marchal [13].

A very well studied example of permutations given by their signatures are the so-called alternating (or zig-zag, or down-up) permutations (see [16] for a survey). Their signatures belong to the language expressed by the regular expression  $(\mathbf{da})^*(\mathbf{d} + \varepsilon)$  (in other words they satisfy  $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4$ ...).

To a language  $L \subseteq {\mathbf{a}, \mathbf{d}}^*$ , we associate the class  $\mathbf{sg}^{-1}(L)$  of permutations whose signature is in L. Many classes of permutations can be expressed in that way (e.g. alternating permutations, those with an even number of descents).

We state and address the two problems of counting and randomly generating in  $\mathbf{sg}^{-1}(L)$  when the language of signatures L is regular. We propose Algorithm 1 that returns a closed form formula for the exponential generating function (EGF) of  $\mathbf{sg}^{-1}(L)$ . That is a formal power series  $\sum a_n \frac{z^n}{n!}$  where the  $n^{th}$  coefficient  $a_n$ counts the permutations of length n with signature in L. With such an EGF, it is easy to recover the number  $a_n$  and some estimation of the growth rate of  $a_n$  (see [9] for an overview of analytic combinatorics). The random generation is done by

<sup>\*</sup> This research is supported in part by ERC Advanced Grant VERIWARE and was also supported by the ANR project EQINOCS (ANR-11-BS02-004).

<sup>\*\*</sup> Omitted proofs and detailed examples can be found in Chapter 8 of [4].

an algorithm described in Theorem 3. The regular language L together with n the size of permutation to generate are the inputs while the outputs are n-length random permutations with signatures in L equally probable to be returned.

Timed automata were introduced in [1] to model and verify properties of real-time systems. Our theory is based on a geometric interpretation of timed languages recognized by timed automata initiated in [3]. In that paper the authors introduced the concept of volume and entropy of timed languages. With these authors we defined and characterized volume generating function of timed language in [2]. In this latter paper a link between enumerative combinatorics and timed languages was foreseen. Here we establish such a link. The passage from a class of permutations to a timed language is in two steps. First we associate order and chain polytopes to signatures which are particular cases of Stanley's poset polytopes [15]. Then we interpret the chain polytopes of a signature w as the set of delays which together with w forms a timed word of a well chosen timed language.

**Related works.** Particular regular languages of signatures are considered in [7] under the name of consecutive descent pattern avoidance. Numerous other works treat more general cases of (consecutive) pattern avoidance (see [8], [12]) and are quite incomparable to our work. Indeed, certain classes of permutations avoiding a finite set of patterns cannot be described as a language of signatures while some classes of permutations involving regular languages cannot be described by finite pattern avoidance (e.g. the permutations with an even number of descents).

The random sampler of timed words (Algorithm 2) is an adaptation to the timed case of the so-called recursive method of [14] developed by [10]. It has been improved for the particular case of generation of words in regular languages [5].

Further connections to related works are considered at the end of section 4.

## 1 Two problem statements

All along the paper we use the two letter alphabet  $\{\mathbf{a}, \mathbf{d}\}\$  whose elements must be read as "ascent" and "descent". Words of  $\{\mathbf{a}, \mathbf{d}\}\$  are called *signatures*. For  $n \in \mathbb{N}, [n]$  denotes  $\{1, \ldots, n\}$  and  $\mathfrak{S}_n$  the set of permutations of [n]. We use the one line notation, for instance  $\sigma = 231$  means that  $\sigma(1) = 2, \sigma(2) = 3, \sigma(3) = 1$ .

Let *n* be a positive integer. The *signature* of a permutation  $\sigma = \sigma_1 \cdots \sigma_n$  is the word  $u = u_1 \cdots u_{n-1} \in \{\mathbf{a}, \mathbf{d}\}^{n-1}$  denoted by  $\mathbf{sg}(\sigma)$  such that for  $i \in [n]$ ,  $\sigma_i < \sigma_{i+1}$  iff  $u_i = \mathbf{a}$  (we speak of an "ascent") and  $\sigma_i > \sigma_{i+1}$  iff  $u_i = \mathbf{d}$  (we speak of a "descent"), for instance  $\mathbf{sg}(21354) = \mathbf{sg}(32451) = \mathbf{daad}$ .

This notion appears in the literature under several different names and forms such as descent word, descent set, ribbon diagram, etc. We are interested in  $\mathbf{sg}^{-1}(L) = \{\sigma \mid \mathbf{sg}(\sigma) \in L\}$ : the class of permutations with signature in  $L \subseteq \{\mathbf{a}, \mathbf{d}\}^*$ . Given a language L we denote by  $L_n$  the sub-language of L restricted to its *n*-length words. The exponential generating function of  $\mathbf{sg}^{-1}(L)$  is

$$F_L(z) =_{\text{def}} \sum_{\sigma \in \text{sg}^{-1}(L)} \frac{z^{|\sigma|}}{|\sigma|!} = \sum_{n \ge 1} |\text{sg}^{-1}(L_{n-1})| \frac{z^n}{n!}.$$



Fig. 1. From left to right: automata for  $L^{ex}$ ,  $L^{ex'}$  and  $st_d(L^{ex'})$ 

*Example 1.* Consider as a running example the class of "up-up-down-down" permutations with signature in the language<sup>1</sup>  $L^{ex} = (\mathbf{aadd})^*(\mathbf{aa} + \varepsilon)$  recognized by the automaton depicted in the left of Figure 1. The theory developed in the paper permits to find the exponential generating function of  $\mathbf{sg}^{-1}(L^{ex})$ :

$$F_{L^{ex}}(z) = \frac{\sinh(z) - \sin(z) + \sin(z)\cosh(z) + \sinh(z)\cos(z)}{1 + \cos(z)\cosh(z)}.$$

Its Taylor expansion is

$$z + \frac{z^3}{3!} + 6\frac{z^5}{5!} + 71\frac{z^7}{7!} + 1456\frac{z^9}{9!} + 45541\frac{z^{11}}{11!} + 2020656\frac{z^{13}}{13!} + \dots$$

For instance, there are 1456 up-up-down-down permutations of length 9.

Now we state the two problems solved in this paper.

Problem 1. Design an algorithm which takes as input a regular language  $L \subseteq {\mathbf{a}, \mathbf{d}}^*$  and returns a closed form formula for  $F_L(z)$ .

Problem 2. Design an algorithm which takes as input a regular language  $L \subseteq \{\mathbf{a}, \mathbf{d}\}^*$  and  $n \geq 1$  and returns a random permutation  $\sigma$  uniformly in  $\mathbf{sg}^{-1}(L_{n-1})$ , that is the probability for each  $\sigma \in \mathbf{sg}^{-1}(L_{n-1})$  to be returned is  $1/|\mathbf{sg}^{-1}(L_{n-1})|$ .

## 2 A timed and geometric approach

In section 2.1 we recall definition of order and chain polytopes associated to signatures. We introduce a sequence of sets  $\mathcal{O}_n(L) \subseteq [0,1]^n$  and see how the two problems posed can be reformulated as computing the volume generating function of this sequence and generating points uniformly in  $\mathcal{O}_n(L)$ . Then we define a timed language  $\mathbb{L}'$  associated to L as well as its volume sequence (section 2.2) and describe a volume preserving transformation between  $\mathcal{O}_n(L)$  and  $\mathbb{L}'_n$ .

#### 2.1 Order and chain polytopes of signatures.

We say that a collection of polytopes  $(S_1, \dots, S_n)$  is an *almost disjoint partition* of a set A if it is the union of  $S_i$  and they have pairwise a null volume intersection. In this case we write  $S = \bigsqcup_{i=1}^{n} S_i$ .

 $<sup>^{1}</sup>$  We identify regular expressions with the regular languages they express.

The set  $\{(\nu_1, \ldots, \nu_n) \in [0, 1]^n \mid 0 \leq \nu_{\sigma_1^{-1}} \leq \ldots \leq \nu_{\sigma_n^{-1}} \leq 1\}$  is called the order simplex<sup>2</sup> of  $\sigma$  and denoted by  $\mathcal{O}(\sigma)$ . For instance  $\boldsymbol{\nu} = (0.3, 0.2, 0.4, 0.5, 0.1)$  belongs to  $\mathcal{O}(32451)$  since  $\nu_5 \leq \nu_2 \leq \nu_1 \leq \nu_3 \leq \nu_4$  and  $(32451)^{-1} = 52134$ . The set  $O(\sigma)$  for  $\sigma \in \mathfrak{S}_n$  forms an almost disjoint partition of  $[0, 1]^n$ . By symmetry all the order simplices of permutations have the same volume which is 1/n!.

If  $\boldsymbol{\nu}$  is uniformly sampled in  $[0,1]^n$  then it falls in any  $O(\sigma)$  with probability 1/n!. To retrieve  $\sigma$  from  $\boldsymbol{\nu}$  it suffices to use a sorting algorithm. We denote by  $\Pi(\boldsymbol{\nu})$  the permutation  $\sigma$  returned by the sorting algorithm on  $\boldsymbol{\nu}$ , that is such that  $0 \leq \nu_{\sigma_1^{-1}} \leq \ldots \leq \nu_{\sigma_n^{-1}} \leq 1$ . Moreover with probability 1,  $\boldsymbol{\nu}$  has pairwise distinct<sup>3</sup>. coordinates and one can define its *signature*  $sg(\boldsymbol{\nu}) = u_1 \ldots u_{n-1}$  by  $u_i = \mathbf{a}$  if  $\nu_i < \nu_{i+1}$  and  $u_i = \mathbf{d}$  if  $\nu_i > \nu_{i+1}$ . For instance  $sg(0.3, 0.2, 0.4, 0.5, 0.1) = \mathbf{daad}$ .

The order polytope  $\mathcal{O}(u)$  [15] of a signature  $u \in \{\mathbf{a}, \mathbf{d}\}^{n-1}$  is the set of vectors  $\boldsymbol{\nu}$  such that for all  $i \leq n-1$ , if  $u_i = \mathbf{a}$  then  $\nu_i \leq \nu_{i+1}$  and  $\nu_i \geq \nu_{i+1}$  otherwise. That is the topological closure of  $\{\boldsymbol{\nu} \in [0,1]^n \mid \mathsf{sg}(\boldsymbol{\nu}) = u\}$ . It is clear that the collection of order simplices  $\mathcal{O}(\sigma)$  with all  $\sigma$  having the same signature u form an almost disjoint partition of the order polytope  $\mathcal{O}(u)$ :  $\mathcal{O}(u) = \bigsqcup_{\sigma \in \mathsf{sg}^{-1}(u)} \mathcal{O}(\sigma)$ , for instance  $\mathcal{O}(\mathsf{daa}) = \mathcal{O}(2134) \sqcup \mathcal{O}(3124) \sqcup \mathcal{O}(4123)$ . Passing to volume we get:

$$\operatorname{Vol}(\mathcal{O}(u)) = \sum_{\sigma \in \operatorname{sg}^{-1}(u)} \operatorname{Vol}(\mathcal{O}(\sigma)) = \frac{|\operatorname{sg}^{-1}(u)|}{n!}.$$
 (1)

Let L be a language of signatures and  $n \geq 1$ , then the family  $(\mathcal{O}(u))_{u \in L_{n-1}}$ forms an almost disjoint partition of a subset of  $[0,1]^n$  called the  $n^{th}$  order set of L and denoted by  $\mathcal{O}_n(L)$ :

$$\mathcal{O}_n(L) = \bigsqcup_{u \in L_{n-1}} \mathcal{O}(u) = \bigsqcup_{\sigma \in \mathsf{sg}^{-1}(L_{n-1})} \mathcal{O}(\sigma) = \overline{\{\boldsymbol{\nu} \in [0,1]^n \mid \mathsf{sg}(\boldsymbol{\nu}) \in L_{n-1}\}}.$$
 (2)

For volumes we get:

$$\operatorname{Vol}(\mathcal{O}_n(L)) = \sum_{u \in L_{n-1}} \operatorname{Vol}(\mathcal{O}(u)) = \sum_{\sigma \in \operatorname{sg}^{-1}(L_{n-1})} \operatorname{Vol}(\mathcal{O}(\sigma)) = \frac{|\operatorname{sg}^{-1}(L_{n-1})|}{n!} \quad (3)$$

The chain polytope [15] of a signature u is the set  $\mathcal{C}(u)$  of vectors  $\mathbf{t} \in [0,1]^n$  such that for all  $i < j \leq n$  and  $l \in \{\mathbf{a}, \mathbf{d}\}, w_i \cdots w_{j-1} = l^{j-i} \Rightarrow t_i + \ldots + t_j \leq 1$ .

*Example 2.* A vector  $(t_1, t_2, t_3, t_4, t_5) \in [0, 1]^5$  belongs to  $\mathcal{C}(\mathbf{daad})$  iff  $t_1 + t_2 \leq 1, t_2 + t_3 + t_4 \leq 1, t_4 + t_5 \leq 1$  iff  $1 - t_1 \geq t_2 \leq t_2 + t_3 \leq 1 - t_4 \geq t_5$  iff  $(1 - t_1, t_2, t_2 + t_3, 1 - t_4, t_5) \in \mathcal{O}(\mathbf{daad})$ .

More generally, for w = ul with  $u \in \{\mathbf{a}, \mathbf{d}\}^*$ ,  $l \in \{\mathbf{a}, \mathbf{d}\}$  and n = |w|, there is a volume preserving transformation  $(t_1, \dots, t_n) \mapsto (\nu_1, \dots, \nu_n)$  from the chain polytope  $\mathcal{C}(u)$  to the order polytope  $\mathcal{O}(u)$  defined as follows.

<sup>&</sup>lt;sup>2</sup> Order simplices, order and chain polytopes of signatures defined here are particular cases of Stanley's order and chain polytopes of posets [15].

<sup>&</sup>lt;sup>3</sup> Alternatively  $sg(\nu) =_{def} sg(\Pi(\nu))$  (defined also when some coordinates are equal).

Let  $j \in [n]$  and i be the index such that  $w_i \cdots w_{j-1}$  is a maximal ascending or descending block, that is i is minimal such that  $w_i \cdots w_{j-1} = l^{j-i}$  with  $l \in \{\mathbf{a}, \mathbf{d}\}^*$ . If  $w_j = \mathbf{d}$  we define  $\nu_j = 1 - \sum_{k=i}^j t_k$  and  $\nu_j = \sum_{k=i}^j t_k$  otherwise.

**Proposition 1 (simple case of Theorem 2.1 of [11]).** The mapping  $\phi_{ul}$ :  $(t_1, \dots, t_n) \mapsto (\nu_1, \dots, \nu_n)$  is a volume preserving transformation from  $\mathcal{C}(u)$  to  $\mathcal{O}(u)$ . It can be computed in linear time using the following recursive definition:  $|\nu_i = \nu_{i-1} + t_i$  if  $w_{i-1}w_i = \mathbf{aa}$ :

$$\begin{vmatrix} \nu_1 = t_1 & \text{if } w_1 = \mathbf{a} \\ \nu_1 = 1 - t_1 & \text{if } w_1 = \mathbf{d} \end{vmatrix} \text{ and for } i \ge 2: \begin{vmatrix} \nu_i = \nu_{i-1} + t_i & \text{if } w_{i-1}w_i = \mathbf{da}, \\ \nu_i = t_i & \text{if } w_{i-1}w_i = \mathbf{da}, \\ \nu_i = 1 - t_i & \text{if } w_{i-1}w_i = \mathbf{da}, \\ \nu_i = 1 - t_i & \text{if } w_{i-1}w_i = \mathbf{da}, \\ \nu_i = \nu_{i-1} - t_i & \text{if } w_{i-1}w_i = \mathbf{da}, \end{vmatrix}$$

As a corollary of (3) and Proposition 1 the first problem can be reformulated in geometric terms as follows.

**Corollary 1.** For every  $L \in {\mathbf{a}, \mathbf{d}}^*$  the following equalities hold:

$$F_L(z) = \sum_{n \geq 1} \operatorname{Vol}(\mathcal{O}_n(L)) z^n = \sum_{u \in L} \operatorname{Vol}(\mathcal{O}(u)) z^{|u|-1} = \sum_{u \in L} \operatorname{Vol}(\mathcal{C}(u)) z^{|u|-1}.$$

For the second problem, it suffices to generate uniformly a vector  $\boldsymbol{\nu} \in \mathcal{O}_n(L)$ and then sort it to get a permutation  $\sigma = \Pi(\boldsymbol{\nu})$ . As the simplices  $\mathcal{O}(\sigma)$  for  $\sigma \in \mathbf{sg}^{-1}(L_n)$  form an almost disjoint partition of  $\mathcal{O}_n(L)$  and all these simplices have the same volume 1/n!, they are equally probable to receive the random vector  $\boldsymbol{\nu}$ . Hence all  $\sigma \in \mathbf{sg}^{-1}(L_n)$  have the same probability to be chosen.

In fact, it is not clear how to fit the sequence of order sets (when n varies) with the dynamics of the language L. We prefer to use a timed language for which we can write recursive equations on volumes (inspired by [3,2]). The reduction from the sequence of order sets to the timed language is mainly given by Proposition 1 since this latter language is a formal union of chain polytopes (Proposition 2).

## 2.2 Timed semantics of a language of signatures: $(\mathbb{L}'_n)_{n \in \mathbb{N}}$

This section is inspired by timed automata theory and designed for non experts. We adopt a non standard<sup>4</sup> and self-contained approach based on the notion of clock languages introduced by [6] and used in our previous work [2].

Timed languages, their volumes and their generating functions An alphabet of *timed events* is the product  $\mathbb{R}^+ \times \Sigma$  where  $\Sigma$  is a finite alphabet. The meaning of a timed event  $(t_i, w_i)$  is that  $t_i$  is the *time delay* before the *event*  $w_i$ . A *timed word* is just a word of timed events and a *timed language* a set of timed words. Adopting a geometric point of view, a timed word is a vector of delays  $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{R}^n$  together with a word of events  $w = w_1 \cdots w_n \in \Sigma^n$ . That is why we sometimes write such a timed word  $(\mathbf{t}, w)$  instead of  $(t_1, w_1) \cdots (t_n, w_n)$ . With this convention, given a timed language  $\mathbb{L}' \subseteq (\mathbb{R}^+ \times \Sigma)^*$ , its restriction

 $<sup>^{4}</sup>$  We refer the reader to [1] for a standard approach of timed automata theory.

to *n*-length words  $\mathbb{L}'_n$  can be seen as a formal union of sets  $\biguplus_{w \in \Sigma^n} \mathbb{L}'_w \times \{w\}$ where  $\mathbb{L}'_w = \{\mathbf{t} \in \mathbb{R}^n \mid (\mathbf{t}, w) \in \mathbb{L}'\}$  is the set of delay vectors that together with *w* form a timed word of  $\mathbb{L}'$ . In the sequel we will only consider languages  $\mathbb{L}'$  for which every  $\mathbb{L}'_w$  is volume measurable. To such  $\mathbb{L}'_n$  one can associate a sequence of volumes and a *volume generating function* as follows:

$$\operatorname{Vol}(\mathbb{L}'_n) = \sum_{w \in \varSigma^n} \operatorname{Vol}(\mathbb{L}'_w); \quad VGF(\mathbb{L}')(z) = \sum_{w \in \varSigma^*} \operatorname{Vol}(\mathbb{L}'_w) z^{|w|} = \sum_{n \in \mathbb{N}} \operatorname{Vol}(\mathbb{L}'_n) z^n = \sum_{w \in \varSigma^*} \operatorname{Vol}(\mathbb{L}'_w) z^{|w|} = \sum_{w \in \boxtimes} \operatorname{Vol}(\mathbb{L}'_w) z^{|w|$$

The clock semantics of a signature. A *clock* is a non-negative real variable. Here we only consider two clocks bounded by 1 and denoted by  $x^{\mathbf{a}}$  and  $x^{\mathbf{d}}$ . A *clock* word is a tuple whose component are a starting clock vector  $(x_0^{\mathbf{a}}, x_0^{\mathbf{d}}) \in [0, 1]^2$ , a timed word  $(t_1, a_1) \cdots (t_n, a_n) \in ([0, 1] \times \{\mathbf{a}, \mathbf{d}\})^*$  and an ending clock vector  $(x_n^{\mathbf{a}}, x_n^{\mathbf{d}}) \in [0, 1]^2$ . It is denoted by  $(x_0^{\mathbf{a}}, x_0^{\mathbf{d}}) \xrightarrow{(t_1, a_1) \cdots (t_n, a_n)} (x_n^{\mathbf{a}}, x_n^{\mathbf{d}})$ . Two clock words  $\mathbf{x}_0 \xrightarrow{\mathbf{w}} \mathbf{x}_1$  and  $\mathbf{x}_2 \xrightarrow{\mathbf{w}'} \mathbf{x}_3$  are said to be compatible if  $\mathbf{x}_2 = \mathbf{x}_1$ , in this case their product is  $(\mathbf{x}_0 \xrightarrow{\mathbf{w}} \mathbf{x}_1) \cdot (\mathbf{x}_2 \xrightarrow{\mathbf{w}'} \mathbf{x}_3) = \mathbf{x}_0 \xrightarrow{\mathbf{w}\mathbf{w}'} \mathbf{x}_3$ . A *clock language* is a set of clock words. The product of two clock languages  $\mathcal{L}$  and  $\mathcal{L}'$  is

$$\mathcal{L} \cdot \mathcal{L}' = \{ c \cdot c' \mid c \in \mathcal{L}, \ c' \in \mathcal{L}', \ c \text{ and } c' \text{ compatible} \}.$$
(4)

The clock language<sup>5</sup>  $\mathcal{L}(\mathbf{a})$  (resp.  $\mathcal{L}(\mathbf{d})$ ) of an ascent (resp. a descent) is the set of clock words of the form  $(x^{\mathbf{a}}, x^{\mathbf{d}}) \xrightarrow{(t, \mathbf{a})} (x^{\mathbf{a}} + t, 0)$  (resp.  $(x^{\mathbf{a}}, x^{\mathbf{d}}) \xrightarrow{(t, \mathbf{d})} (0, x^{\mathbf{d}} + t)$ ) and such that  $x^{\mathbf{a}} + t \in [0, 1]$  and  $x^{\mathbf{d}} + t \in [0, 1]$  (and by definition of clocks and delays  $x^{\mathbf{a}} \geq 0$ ,  $x^{\mathbf{d}} \geq 0$ ,  $t \geq 0$ ). These definitions extend inductively to all signatures:  $\mathcal{L}(u_1 \cdots u_n) = \mathcal{L}(u_1) \cdots \mathcal{L}(u_n)$  (with product (4)).

$$\begin{array}{l} Example \ 3. \ (0,0) \xrightarrow{(0.7,\mathbf{d})(0.2,\mathbf{a})(0.2,\mathbf{a})(0.5,\mathbf{d})} (0,0.5) \in \mathcal{L}(\mathbf{daad}) \text{ since} \\ (0,0) \xrightarrow{(0.7,\mathbf{d})} (0,0.7) \in \mathcal{L}(\mathbf{d}); \qquad (0,0.7) \xrightarrow{(0.2,\mathbf{a})} (0.2,0) \in \mathcal{L}(\mathbf{a}); \\ (0.2,0) \xrightarrow{(0.2,\mathbf{a})} (0.4,0) \in \mathcal{L}(\mathbf{a}); \qquad (0.4,0) \xrightarrow{(0.5,\mathbf{a})} (0,0.5) \in \mathcal{L}(\mathbf{d}). \end{array}$$

The timed semantics of a language of signatures. The timed polytope associated to a signature  $w \in \{a, d\}^*$  is

$$P_w =_{\text{def}} \{ \boldsymbol{t} \mid (0,0) \xrightarrow{(\boldsymbol{t},w)} \boldsymbol{y} \in \mathcal{L}(w) \text{ for some } \boldsymbol{y} \in [0,1]^2 \}.$$

For instance  $(0.7, 0.2, 0.2, 0.5, 0.1) \in P_{daada}$ . The timed semantics of a language of signatures L' is

$$\mathbb{L}' = \{(\boldsymbol{t}, w) \mid \boldsymbol{t} \in P_w \text{ and } w \in L'\} = \bigcup_{w \in L'} P_w \times \{w\}.$$

This language restricted to words of length n is  $\mathbb{L}'_n = \bigcup_{w \in L'_n} P_w \times \{w\}$ , its volume is  $\operatorname{Vol}(\mathbb{L}'_n) = \sum_{w \in L'} \operatorname{Vol}(P_w)$ .

<sup>&</sup>lt;sup>5</sup> A reader acquainted with timed automata would have noticed that the clock language  $\mathcal{L}(\mathbf{a})$  (resp.  $\mathcal{L}(\mathbf{d})$ ) corresponds to a transition of a timed automaton where the guards  $x^{\mathbf{a}} \leq 1$  and  $x^{\mathbf{d}} \leq 1$  are satisfied and where  $x^{\mathbf{d}}$  (resp.  $x^{\mathbf{a}}$ ) is reset.

The link with order and chain polytopes of signatures. We first state the link between timed polytopes and chain polytopes.

**Proposition 2.** Given a word  $u \in \{\mathbf{a}, \mathbf{d}\}^*$  and  $l \in \{\mathbf{a}, \mathbf{d}\}$ , the timed polytope of ul is the chain polytope of  $u: P_{ul} = C(u)$ .

Hence Proposition 1 links the timed polytope  $P_{ul} = \mathcal{C}(u)$  of a signature of length n + 1 and the order polytopes  $\mathcal{O}(u)$  of a signature of length n. We correct the mismatch of length using prolongation of languages. A language L' is called a *prolongation* of a language L whenever the truncation of the last letter  $w_1 \dots w_n \mapsto w_1 \dots w_{n-1}$  is a bijection between L' and L. Every language Lhas prolongations, for instance L' = Ll for  $l \in \{\mathbf{a}, \mathbf{d}\}$ . A prolongation of  $L^{ex}$  is  $L^{ex'} = (\mathbf{aadd})^*(\mathbf{aad} + \mathbf{a})$  recognized by the automaton depicted in the middle of Figure 1. Proposition 1 can be extended to language of signatures as follows.

**Corollary 2.** Let  $L \subseteq \{\mathbf{a}, \mathbf{d}\}^*$  and  $\mathbb{L}'$  be the timed semantics of a prolongation of L then for all  $n \in \mathbb{N}$ , the following function is a volume preserving transformation between  $\mathbb{L}'_n$  and  $\mathcal{O}_n(L)$ . Moreover it is computable in linear time.

$$\begin{aligned} \phi : & \mathbb{L}'_n \to \mathcal{O}_n(L) \\ & (\boldsymbol{t}, w) \mapsto \phi_w(\boldsymbol{t}) \end{aligned}$$
 (5)

As a consequence, the two problems can be solved if we know how to compute the VGF of a timed language  $\mathbb{L}'$  and how to generate timed vector uniformly in  $\mathbb{L}'_n$ . A characterization of the VGF of a timed language as a solution of a system of differential equations is done in [2]. Nevertheless the equations of this article are quite uneasy to handle and don't give a closed form formula for the VGF. To get simpler equations than in [2] we work with a novel class of timed languages involving two kinds of transitions S and T.

## 2.3 The S-T (timed) language encoding.

The S-T-encoding We consider the finite alphabet {S, T} whose elements must be respectively read as *straight* and *turn*. The S-T-encoding of type  $l \in \{\mathbf{a}, \mathbf{d}\}$ of a word  $w \in \{\mathbf{a}, \mathbf{d}\}^*$  is a word  $w' \in \{\mathbf{S}, \mathbf{T}\}^*$  denoted by  $\mathtt{st}_l(w)$  and defined recursively as follows: for every  $i \in [n]$ ,  $w'_i = \mathtt{S}$  if  $w_i = w_{i-1}$  and  $w'_i = \mathtt{T}$  otherwise, with the convention that  $w_0 = l$ . The mapping  $\mathtt{st}_l$  is invertible and can also be defined recursively. Indeed  $w = \mathtt{st}_l^{-1}(w')$  iff for every  $i \in [n]$ ,  $w_i = w_{i-1}$ if  $w'_i = \mathtt{S}$  and  $w_i \neq w_{i-1}$  otherwise, with convention that  $w_0 = l$ . Notion of S-T-encoding can be extended naturally to languages. For the running example:  $\mathtt{st}_d(L^{ex'}) = (\mathtt{TS})^*\mathtt{T}$ . We call an S-T-automaton, a deterministic finite state automaton with transition alphabet {S, T} (see Figure 1 for an S-T-automaton recognizing  $\mathtt{st}_d(L^{ex'})$ ).

Timed semantics and S-T-encoding In the following we define clock and timed languages similarly to what we have done in section 2.2. Here we need only one clock x that remains bounded by 1. We define the clock language associated

to **S** by  $\mathcal{L}(\mathbf{S}) = \{x \xrightarrow{(t,\mathbf{S})} x + t \mid x \in [0,1], t \in [0,1-x]\}$  and the clock language associated to **T** by  $\mathcal{L}(\mathbf{T}) = \{x \xrightarrow{(t,\mathbf{T})} t \mid x \in [0,1], t \in [0,1-x]\}$ . Let  $L'' \subseteq \{\mathbf{S},\mathbf{T}\}^*$ we denote by L''(x) the timed language starting from  $x: L''(x) = \{(t,w) \mid \exists y \in [0,1], x \xrightarrow{(t,w)} y \in \mathcal{L}(w), w \in L''\}$ . The timed semantics of  $L'' \subseteq \{\mathbf{S},\mathbf{T}\}^*$  is L''(0).

The S-T-encodings yields a natural volume preserving transformation between timed languages:

**Proposition 3.** Let  $L' \subseteq \{\mathbf{a}, \mathbf{d}\}^*$ ,  $l \in \{\mathbf{a}, \mathbf{d}\}$ ,  $\mathbb{L}'$  be the timed semantics of L'and  $\mathbb{L}''$  be the timed semantics of  $st_l(L')$  then the function  $(t, w) \mapsto (t, st_l^{-1}(w))$ is a volume preserving transformation from  $\mathbb{L}''_n$  to  $\mathbb{L}'_n$ .

Using notation and results of Corollary 2 and Proposition 3 we get a volume preserving transformation from  $\mathbb{L}''_n$  to  $\mathcal{O}_n(L)$ .

**Theorem 1.** The function  $(t, w) \mapsto \phi_{st_l^{-1}(w)}(t)$  is a volume preserving transformation from  $\mathbb{L}''_n$  to  $\mathcal{O}_n(L)$  computable in linear time. In particular

$$\operatorname{Vol}(\mathbb{L}''_n) = \frac{|\operatorname{sg}^{-1}(L_{n-1})|}{n!} \text{ for } n \ge 1 \text{ and } VGF(\mathbb{L}'')(z) = F_L(z).$$

Thus to solve Problem 1 it suffices to characterize the VGF of an S-T-automaton.

## 3 Solving the two problems

#### 3.1 Characterization of the VGF of an S-T-automaton.

In this section we characterize precisely the VGF of the timed language recognized by an S-T-automaton. This solves Problem 1.

We have defined just above timed language L''(x) parametrized by an initial clock x. Given an S-T-automaton, we can also consider the initial state p as a parameter and write Kleene like systems of equations on parametric language  $L_p(x)$  (similarly to [2]). More precisely, let  $\mathcal{A} = (\{\mathsf{S},\mathsf{T}\}, Q, q_0, F, \delta)$  be an S-T-automaton with states Q, initial state  $q_0 \in Q$ , final states  $F \subseteq Q$  and transition function  $\delta : Q \times \{\mathsf{S},\mathsf{T}\} \to Q$ . To every state  $p \in Q$  we denote by  $L_p \subseteq \{\mathsf{S},\mathsf{T}\}^*$  the language starting from p that is recognized by  $\mathcal{A}_p =_{def} \{\{\mathsf{S},\mathsf{T}\}, Q, p, F, \delta\}$ . Then for every  $p \in Q$ , we have a parametric language equation:

$$L_p(x) = \left[ \bigcup_{t \le 1-x} (t, \mathbf{S}) L_{\delta(p, \mathbf{S})}(x+t) \right] \cup \left[ \bigcup_{t \le 1-x} (t, \mathbf{T}) L_{\delta(p, \mathbf{T})}(t) \right] \cup (\varepsilon \text{ if } p \in F).$$
(6)

We denote by  $f_p(x, z)$  and  $V_p(z)$  the volume generating function of  $L_p(x)$  and  $L_p$  respectively and are interested in  $V_{q_0}(z)$ . As in [2], we pass from equation on languages (6) to equation on generating functions:

$$f_p(x,z) = z \int_x^1 f_{\delta(p,\mathbf{S})}(s,z) ds + z \int_0^{1-x} f_{\delta(p,\mathbf{T})}(t,z) dt + (1 \text{ if } p \in F)$$
(7)

In matrix notation:

$$\boldsymbol{f}(x,z) = zM_{\rm S} \int_x^1 \boldsymbol{f}(s,z) ds + zM_{\rm T} \int_0^{1-x} \boldsymbol{f}(t,z) dt + \boldsymbol{F}$$
(8)

where  $\mathbf{f}(x, z), \int_x^1 \mathbf{f}(s, z) ds$  and  $\int_0^{1-x} \mathbf{f}(t, z) dt$  are the column vectors whose coordinates are respectively the  $f_p(x, z), \int_x^1 f_p(s, z) ds$  and  $\int_0^{1-x} f_p(t, z) dt$  for  $p \in Q$ . The  $p^{th}$  coordinate of the column vector  $\mathbf{F}$  is 1 if  $p \in F$  and 0 otherwise. The  $Q \times Q$ -matrices  $M_{\mathsf{S}}$  and  $M_{\mathsf{T}}$  are the adjacency matrices corresponding to letter  $\mathsf{S}$  and  $\mathsf{T}$  that is for  $l \in \{\mathsf{S},\mathsf{T}\}, M_l(p,q) = 1$  if  $\delta(p,l) = q$  and 0 otherwise.

The equation (8) is equivalent to the differential equation:

$$\frac{\partial}{\partial x}\boldsymbol{f}(x,z) = -zM_{\rm S}\boldsymbol{f}(x,z) - zM_{\rm T}\boldsymbol{f}(1-x,z) \tag{9}$$

with boundary condition

$$\boldsymbol{f}(1,z) = \boldsymbol{F}.\tag{10}$$

The equation (9) is equivalent to the following linear homogeneous system of ordinary differential equations with constant coefficients:

$$\frac{\partial}{\partial x} \begin{pmatrix} \boldsymbol{f}(x,z) \\ \boldsymbol{f}(1-x,z) \end{pmatrix} = z \begin{pmatrix} -M_{\rm S} - M_{\rm T} \\ M_{\rm T} & M_{\rm S} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}(x,z) \\ \boldsymbol{f}(1-x,z) \end{pmatrix}$$
(11)

whose solution is of the form

$$\begin{pmatrix} \boldsymbol{f}(x,z) \\ \boldsymbol{f}(1-x,z) \end{pmatrix} = \exp\left[xz \begin{pmatrix} -M_{\rm S} & -M_{\rm T} \\ M_{\rm T} & M_{\rm S} \end{pmatrix}\right] \begin{pmatrix} \boldsymbol{f}(0,z) \\ \boldsymbol{f}(1,z) \end{pmatrix}.$$
 (12)

Taking x = 1 in (12) and using the boundary condition (10) we obtain:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{V}(z) \end{pmatrix} = \exp\left[z \begin{pmatrix} -M_{\rm S} & -M_{\rm T} \\ M_{\rm T} & M_{\rm S} \end{pmatrix}\right] \begin{pmatrix} \mathbf{V}(z) \\ \mathbf{F} \end{pmatrix}$$
(13)

where V(z) is the vector whose coordinates are  $V_q(z)$  for  $q \in Q$ . Hence,

$$\boldsymbol{F} = A_1(z)\boldsymbol{V}(z) + A_2(z)\boldsymbol{F}; \quad \boldsymbol{V}(z) = A_3(z)\boldsymbol{V}(z) + A_4(z)\boldsymbol{F}$$
(14)

where  $\begin{pmatrix} A_1(z) & A_2(z) \\ A_3(z) & A_4(z) \end{pmatrix} = \exp \left[ z \begin{pmatrix} -M_{\rm S} & -M_{\rm T} \\ M_{\rm T} & M_{\rm S} \end{pmatrix} \right]$ . In particular when z = 0,  $A_1(0) = I - A_3(0) = I$  and thus the two continuous functions  $z \mapsto \det A_1(z)$  and  $z \mapsto \det(I - A_3(z))$  are positive in a neighbourhood of 0. We deduce that the inverses of the matrices  $A_1(z)$  and  $I - A_3(z)$  are well defined in a neighbourhood of 0 and thus both equations of (14) permit to express V(z) wrt. F:

$$\mathbf{V}(z) = [A_1(z)]^{-1} [I - A_2(z)] \mathbf{F}; \quad \mathbf{V}(z) = [I - A_3(z)]^{-1} A_4(z) \mathbf{F}$$
(15)

To sum up, we address Problem 1 with the following theorem.

**Theorem 2.** Given a regular language  $L \subseteq {\mathbf{a}, \mathbf{d}}^*$ , one can compute the exponential generating function  $F_L(z)$  using Algorithm 1.

Some comments about the algorithm. In line 1, several choices are left to the user: the prolongation L' of the language L, the type of the S-T-encoding and the automaton that realizes the S-T-encoding. These choices should be made such that the output automaton has a minimal number of states or more generally such that the matrices  $M_{\rm T}$  and  $M_{\rm S}$  are the simplest possible. Exponentiation of matrices is implemented in most of computer algebra systems.

Algorithm 1 Computation of the generating function

- 1: Compute an S-T-automaton  $\mathcal{A}$  for an extension of L and its corresponding adjacency matrices  $M_{\rm T}$  and  $M_{\rm S}$ ;
- 2: Compute  $\begin{pmatrix} A_1(z) & A_2(z) \\ A_3(z) & A_4(z) \end{pmatrix} =_{def} \exp\left[z \begin{pmatrix} -M_{s} & -M_{T} \\ M_{T} & M_{s} \end{pmatrix}\right];$
- 3: Compute  $V(z) = [A_1(z)]^{-1} [I A_2(z)] \tilde{F}$  (or  $V(z) = [I A_3(z)]^{-1} A_4(z) F$ );

4: return  $V_{q_0}(z)$  the component of V(z) corresponding to the initial state of  $\mathcal{A}$ .

## 3.2 An algorithm for Problem 2

Now we can solve Problem 2 using a uniform sampler of timed words (Algorithm 2), the volume preserving transformation of Theorem 1 and a sorting algorithm.

**Theorem 3.** Let  $L \subseteq {\mathbf{a}, \mathbf{d}}^*$  and  $\mathbb{L}''$  be the timed semantics of a S-T-encoding of type l (for some  $l \in {\mathbf{a}, \mathbf{d}}$ ) of a prolongation of L. The following algorithm permits to achieve a uniform sampling of permutation in  $\mathbf{sg}^{-1}(L_{n-1})$ .

- 1. Choose uniformly an n-length timed word  $(t, w) \in \mathbb{L}_n^{\prime\prime}$  using Algorithm 2;
- 2. Return  $\Pi(\phi_{st_{i}^{-1}(w)}(t))$ .

Uniform sampling of timed words. Recursive formulae (16) and (17) below are freely inspired by those of [3] and of our previous work [2]. They are the key tools to design a uniform sampler of timed word. This algorithm is a lifting from the discrete case of the so-called recursive method (see [5,10]). For all  $q \in Q$ ,  $n \in \mathbb{N}$  and  $x \in [0, 1]$  we denote by  $L_{q,n}(x)$  the language  $L_q(x)$  restricted to *n*length timed words. The languages  $L_{q,n}(x)$  can be recursively defined as follows:  $L_{q,0}(x) = \varepsilon$  if  $q \in F$  and  $L_{q,0} = \emptyset$  otherwise;

$$L_{q,n+1}(x) = \left[ \bigcup_{t \le 1-x} (t, \mathbf{S}) L_{\delta(q, \mathbf{S}), n}(x+t) \right] \cup \left[ \bigcup_{t \le 1-x} (t, \mathbf{T}) L_{\delta(q, \mathbf{T}), n}(t) \right].$$
(16)

For  $q \in Q$  and  $n \geq 0$ , we denote by  $v_{q,n}$  the function  $x \mapsto \operatorname{Vol}[L_{q,n}(x)]$  from [0,1] to  $\mathbb{R}^+$ . Each  $v_{q,n}$  is a polynomial of a degree less or equal to n that can be computed recursively using the recurrent formula:  $v_{q,0}(x) = 1_{q \in F}$  and

$$v_{q,n+1}(x) = \int_{x}^{1} v_{\delta(q,\mathsf{S}),n}(y) dy + \int_{0}^{1-x} v_{\delta(q,\mathsf{T}),n}(y) dy.$$
 (17)

The polynomials  $v_{q,n}(x)$  play a key role for the uniform sampler. They permit also to retrieve directly the terms of the wanted VGF:  $Vol(\mathbb{L}''_n) = v_{q_0,n}(0)$  where  $q_0$  is the initial state of the S-T automaton.

**Theorem 4.** Algorithm 2 is a uniform sampler of timed words of  $\mathbb{L}''_n$ , that is for every volume measurable subset  $A \subseteq \mathbb{L}''_n$ , the probability that the returned timed word belongs to A is  $Vol(A)/Vol(\mathbb{L}''_n)$ .

Algorithm 2 Recursive uniform sampler of timed words

1:  $x_0 \leftarrow 0$ ;  $q_0 \leftarrow$  initial state; 2: for k = 1 to n do Compute  $m_k = v_{q_{k-1}, n-(k-1)}(x_{k-1})$  and  $p_s = \int_{x_{k-1}}^1 v_{\delta(q_{k-1}, s), n-k}(y) dy/m_k$ ; 3:  $b \leftarrow \text{BERNOULLI}(p_s);$  (return 1 with probability  $p_s$  and 0 otherwise) 4: if b = 1 then 5:6:  $w_k \leftarrow \mathbf{S}; q_k \leftarrow \delta(q_{k-1}, \mathbf{S});$ 
$$\begin{split} r &\leftarrow \texttt{RAND}([0,1]); \quad (\text{return a number uniformly sampled in } [0,1]) \\ t_k &\leftarrow \text{the unique solution in } [0,1-x_{k-1}] \text{ of } \frac{1}{m_k p_{\text{S}}} \int_{x_{k-1}}^{x_{k-1}+t_k} v_{q_k,n-k}(y) dy - r = 0; \end{split}$$
7: 8: 9:  $x_k \leftarrow x_{k-1} + t_k;$ 10:else 11:  $w_k \leftarrow \mathsf{T}; q_k \leftarrow \delta(q_{k-1}, \mathsf{T});$  $r \leftarrow \text{RAND}([0,1]);$  (return a number uniformly sampled in [0,1]) 12: $t_k \leftarrow$  the unique solution in  $[0, 1 - x_{k-1}]$  of  $\frac{1}{m_k(1-p_8)} \int_0^{t_k} v_{q_k, n-k}(y) dy - r = 0;$ 13:14:  $x_k \leftarrow t_k;$ 15:end if 16: end for 17: return  $(t_1, w_1)(t_2, w_2) \dots (t_n, w_n)$ 

Some comments about the algorithm. Algorithm 2 requires a precomputation of all functions  $v_{q,k}$  for  $q \in Q$  and  $k \leq n$ . They can be computed in polynomial time by a dynamic programming method using (17). The expressions in lines 8 and 13 are polynomial functions increasing on [x, 1] (the derivative is the integrand which is positive on (x, 1)). Finding the root of such a polynomial can be done numerically and efficiently with a controlled error using a numerical scheme such as the Newton's method. A toy implementation of Algorithm 2 as well as that sketched in Theorem 3 is available on-line http://www.liafa.univ-paris-diderot.fr/~nbasset/sage/sage.htm.

## 4 Discussion, perspectives and related works

We have stated and solved the problems of counting and uniform sampling of permutations with signature in a given regular language of signatures. The timed semantics of such a language is a particular case of regular timed languages (i.e. recognized by timed automata [1]). However, with the approach used, timed languages can be defined from any kind of languages of signatures. A challenging task for us is to treat the case of context free languages. For this we should use as in [2] volume of languages parametrized both by starting and ending states.

Our work can also benefit timed automata research. Indeed, we have proposed a uniform sampler for a particular class of timed languages. An ongoing work is to adapt this algorithm to all deterministic timed automata with bounded clocks using recursive equations of [3].

There is no mention of the parameter x in Algorithm 1. It could be interesting to find a direct explanation of this algorithm (without using parameters). In any case, the parametric approach was crucial in the solution of the second problem. Parametric approaches similar to ours are used in [7,13,17]. In particular recursive equations involving integrals are also described there. Technically, these approaches are based on order polytopes and yield integral operators of the form  $\int_0^x$  and  $\int_x^1$  while ours is based on chain polytopes and yields integral operators  $\int_0^{1-x}$  and  $\int_x^1$ . The fact that these two operators are both null in x = 1 was very useful here. The main novelty is our use of Kleene like equations for regular timed languages and their volume functions (inspired by [2,3]) that allowed us to address the two problems for all regular languages of signatures.

## References

- R. Alur and D. L. Dill. A theory of timed automata. Theor. Comput. Sci., 126(2):183–235, 1994.
- E. Asarin, N. Basset, A. Degorre, and D. Perrin. Generating functions of timed languages. In B. Rovan, V. Sassone, and P. Widmayer, editors, *MFCS*, volume 7464 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2012.
- E. Asarin and A. Degorre. Volume and entropy of regular timed languages: Analytic approach. In J. Ouaknine and F. W. Vaandrager, editors, *FORMATS*, volume 5813 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2009.
- 4. N. Basset. Volumetry of timed languages and applications. PhD thesis, Université Paris-Est, 2013.
- O. Bernardi and O. Giménez. A linear algorithm for the random sampling from regular languages. Algorithmica, 62(1-2):130–145, 2012.
- P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. Journal of Automata, Languages and Combinatorics, 7(2):167–186, 2002.
- R. Ehrenborg and J. Jung. Descent pattern avoidance. Advances in Applied Mathematics, 2012.
- S. Elizalde and M. Noy. Consecutive patterns in permutations. Advances in Applied Mathematics, 30(1):110–125, 2003.
- 9. P. Flajolet and R. Sedgewick. Analytic combinatorics. Camb. Univ. press, 2009.
- P. Flajolet, P. Zimmerman, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1):1– 35, 1994.
- T. Hibi and N. Li. Unimodular equivalence of order and chain polytopes. arXiv preprint arXiv:1208.4029, 2012.
- 12. S. Kitaev. Patterns in permutations and words. Springer, 2011.
- P. Marchal. Generating random permutations with a prescribed descent set. Presentation at Permutation Patterns 2013.
- A. Nijenhuis and H. S. Wilf. Combinatorial algorithms for computers and calculators. Computer Science and Applied Mathematics, New York: Academic Press, 1978, 2nd ed., 1, 1978.
- R. P. Stanley. Two poset polytopes. Discrete & Computational Geometry, 1(1):9– 23, 1986.
- R. P. Stanley. A survey of alternating permutations. In *Combinatorics and graphs*, volume 531 of *Contemp. Math.*, pages 165–196. Amer. Math. Soc., Providence, RI, 2010.
- 17. G. G. Szpiro. The number of permutations with a given signature, and the expectations of their elements. *Discrete Mathematics*, 226(1):423–430, 2001.

## Appendix

We first give extra proof details in the appendix A. We describe the preprocessing of Algorithm 2 and its complexity in the appendix B. Last but not least we details the running example and two others in the appendix C.

## A Some proof details

## **Proof of Proposition 2**

Let w = ul i.e. for all  $i \in [n-1]$   $w_i = u_i$  and  $w_n = l$ .  $P_{ul} \subseteq \mathcal{C}(u)$ ) Let  $(t_1, \ldots, t_n) \in P_w$  i.e. there exist value of clocks  $x_k^a$   $(a \in \{\mathbf{a}, \mathbf{d}\}, k \in [n])$  such that  $x_0^{\mathbf{a}} = x_0^{\mathbf{d}} = 0$  and  $(x_{k-1}^{\mathbf{a}}, x_{k-1}^{\mathbf{d}}) \xrightarrow{(t_k, w_k)} (x_k^{\mathbf{a}}, x_k^{\mathbf{d}}) \in \mathcal{L}(w_k)$ . Let  $i < j \leq n$  and  $a \in \{\mathbf{a}, \mathbf{d}\}$  such that  $w_i \cdots w_{j-1} = a^{j-i}$ , then for  $k \in \{i, \ldots, j-1\}, x_k^a = x_{k-1}^a + t_k$  by definition of  $\mathcal{L}(a)$ . Then  $x_{j-1}^a = x_{i-1}^a + t_i + \ldots + t_{j-1}$ . Moreover  $x_{j-1}^a + t_j \leq 1$  by definition of  $\mathcal{L}(w_j)$  and thus  $t_i + \ldots + t_{j-1} + t_j \leq x_{i-1}^a + t_j \leq 1$  which is the wanted inequality.

 $\mathcal{C}(u) \subseteq P_{ul}$ ) Let  $(t_1, \ldots, t_n) \in \mathcal{C}(u)$ . We show inductively that for every  $a \in \{\mathbf{a}, \mathbf{d}\}$ , the condition  $x_{j-1}^a + t_j \leq 1$  is satisfied and thus that  $x_j^a$  can be defined  $(x_j^a = x_{j-1}^a + t_j \text{ if } w_j = a \text{ and } x_j^a = 0 \text{ otherwise})$ . For this we suppose that clock values  $x_0^a, \ldots, x_{j-1}^a$  are well defined. Let  $lr(x^a, j)$  be the maximal index before transition j such that  $w_{lr(x^a, j)} \neq a$ . Necessarily  $w_{lr(x^a, j)+1} \ldots w_j = a^{j-i}$  and thus  $t_{lr(x^j)+1} + \ldots + t_j \leq 1$  by definition of  $\mathcal{C}(u)$ . This latter sum is equal to  $x_{j-1}^a + t_j \leq 1$  and thus the condition on  $x^a$  imposed by  $\mathcal{L}(u_j)$  is satisfied.  $\Box$ 

## **Proof of Proposition 1**

The function  $\phi_{ul}$  is a volume preserving transformation since it is a linear function given by a unimodular (i.e. an integer matrix having determinant +1 or -1) matrix. Indeed  $\phi_{ul}(\mathbf{t}) = \mathbf{\nu}$  iff  $\mathbf{\nu}^{\top} = M_{ul}\mathbf{t}^{\top} + \mathbf{b}$  with for all  $j \in [n]$ : if  $w_j = \mathbf{a}$ (resp.  $w_j = \mathbf{d}$ ) then the  $j^{th}$  row of the matrix  $M_{ul}$  has only 1s (resp. -1s) between coordinates i and j included and the  $j^{th}$  row of  $\mathbf{b}$  is 0 (resp. -1). One can see that  $M_{ul}$  is upper triangular and has only 1 and -1 on its diagonal and thus is unimodular. Now it remains to prove that  $\mathbf{\nu} (= \phi_{ul}(\mathbf{t}))$  belongs to  $\mathcal{O}(u)$  for  $\mathbf{t} \in \mathcal{C}(u)$ . For this we show that the two conditions (C-1) and (C-2) below are equivalent; the former is the definition of  $(t_1, \dots, t_n) \in \mathcal{C}(u)$  while the latter is equivalent to  $\nu_1, \dots, \nu_n \in \mathcal{O}(u)$ :

- (C-1) for all  $i < j \le n$  and  $l \in \{\mathbf{a}, \mathbf{d}\}, u_i \cdots u_{j-1} = l^{j-i} \Rightarrow t_i + \ldots + t_j \le 1;$
- (C-2) for all  $i < j \le n, u_i \cdots u_{j-1} = \mathbf{a}^{j-i} \Rightarrow \nu_i \le \ldots \le \nu_j \le 1$  and  $u_i \cdots u_{j-1} = \mathbf{d}^{j-i} \Rightarrow \nu_j \le \ldots \le \nu_i \le 1$ .

Let  $i < j \le n$  and  $u_i \cdots u_{j-1} = \mathbf{a}^{j-i}$  then the following chain of inequalities  $[0 \le \nu_i = t_i \le \ldots \le \nu_{j-1} = t_i + \ldots + t_{j-1} \le \nu_j = (1 - t_j \text{ or } t_i + \ldots + t_j) \le 1]$  is equivalent to  $t_i + \ldots + t_j \le 1$ . The case of descents can be proved in a similar way by applying  $x \mapsto 1 - x$  to the preceding inequalities.  $\Box$ 

*Proof.* For all  $\sigma \in \mathbf{sg}_n^{-1}(L)$ , the probability  $p(\sigma)$  that the output is  $\sigma$  is the probability to choose a timed word (t, w) such that  $\Pi[\phi_{\mathtt{st}^{-1}(w)}(t)] = \sigma$ . Since the timed words are uniformly sampled this probability is equal to  $\operatorname{Vol}(\{(t, w) \mid \Pi[\phi_w(t)] = \sigma\})/\operatorname{Vol}(\mathbb{L}''_n)$  which is equal to  $\operatorname{Vol}(\{\nu \mid \Pi(\nu) = \sigma\})/\operatorname{Vol}(\mathbb{L}''_n)$  since the mapping  $(t, w) \mapsto \phi_{\mathtt{st}^{-1}(w)}(t)$  is a volume preserving transformation. The numerator is the volume of the order simplex associated to  $\sigma$  which is  $\operatorname{Vol}(\mathcal{O}(\sigma)) = 1/n!$ ; the denominator  $\operatorname{Vol}(\mathbb{L}''_n)$  is  $|\mathtt{sg}^{-1}(L_{n-1})|/n!$  by virtue of Theorem 1. We get the expected result  $p(\sigma) = (1/n!)/(|\mathtt{sg}^{-1}(L_{n-1})|/n!) = 1/|\mathtt{sg}^{-1}(L_{n-1})|$ . □

#### Sketch of proof of Theorem 4.

One can first check that for all  $k \in [n]$ ,  $(q_{k-1}, x_{k-1}) \xrightarrow{(t_k, w_k)} (q_k, x_k) \in \mathcal{L}(w_k)$ and that  $w_1 \cdots w_n \in L''$ .

We denote by  $p[(t_1, w_1) \cdots (t_n, w_n)]$  the density of probability of the timed word  $(t_1, w_1) \cdots (t_n, w_n) \in \mathbb{L}''$  to be returned. The algorithm is a uniform sampler if it assign the same density of probability to every timed word of  $\mathbb{L}''$ i.e.  $p[(t_1, w_1) \cdots (t_n, w_n)] = 1/\operatorname{Vol}(\mathbb{L}'')$ .

During the  $k^{th}$  loop,  $w_k$  and  $t_k$  are chosen, knowing the preceding general state  $(q_{k-1}, x_{k-1})$  and the index k, according to a density of probability (implicitly defined by the algorithm) denoted by  $p_k[(t_k, w_k) | (q_{k-1}, x_{k-1})]$ . The new general state  $(q_k, x_k)$  is (deterministically) defined using  $(q_{k-1}, x_{k-1})$  and  $(t_k, w_k)$ . The following chain rule is satisfied

$$p[(t_1, w_1) \cdots (t_n, w_n)] = \prod_{k=1}^n p_k[(t_k, w_k) \mid (q_{k-1}, x_{k-1})]$$
(18)

No it suffices to plug (19) proven in Lemma 1 just below in (18) to get the expected result:

$$p[(t_1, w_1) \cdots (t_n, w_n)] = \frac{\prod_{k=1}^n m_{k+1}}{\prod_{k=1}^n m_k} = \frac{m_{n+1}}{m_1} = \frac{v_{q_n,0}(x_n)}{v_{q_0,n}(0)} = \frac{1}{\operatorname{Vol}(\mathbb{L}''_n)}.$$

**Lemma 1.** In Algorithm 2 during the  $k^{th}$  loop for the timed transition  $(t_k, w_k)$  is chosen knowing the current state  $(q_{k-1}, x_{k-1})$  according to the following probability distribution function (variables of the following equation such as  $m_k$  are defined in the algorithm):

$$p_k[(t_k, w_k) \mid (q_{k-1}, x_{k-1})] = \frac{m_{k+1}}{m_k} = \frac{v_{q_k, n-k}(x_k)}{v_{q_{k-1}, n-(k-1)}(x_{k-1})}.$$
 (19)

*Proof.* The choice of  $(t_k, w_k)$  is done in two steps: first  $w_k$  is chosen (and thus  $q_k = \delta(q_{k-1}, w_k)$ ) and then  $t_k$ . We write this

$$p_k[(t_k, w_k) \mid (q_{k-1}, x_{k-1})] = p_k[w_k \mid (q_{k-1}, x_{k-1})]p_k[t_k \mid q_k \text{ and } x_{k-1}]$$
(20)

Remark that b = 1 iff  $w_k = S$  and thus  $p_k[S | (q_{k-1}, x_{k-1})] = p_S$  (the probability that 1 is returned in line 4) and  $p_k[T | (q_{k-1}, x_{k-1})] = 1 - p_S$  otherwise.

In both cases (b = 0 or 1) the delay  $t_k$  is sampled using the so-called inverse transform sampling. This method states that to sample a random variable according to a probability density function (PDF) p(t) (here  $p(t) = p_k[t \mid q_k \text{ and } x_{k-1}]$ ) it suffices to uniformly sample a random number in [0, 1] and define t such that  $\int_0^t p(t')dt' = r$ . The latter integral is known as the cumulative density function<sup>6</sup> (CDF) associated to p.

- When b = 1 (and thus  $w_k = S$ ), the CDF used in the algorithm is

$$t \mapsto \frac{1}{m_k p_{\mathbf{S}}} \int_0^t v_{q_k, n-k}(x_{k-1} + t') dt'.$$

Its corresponding PDF is

$$p_k[t_k \mid q_k \text{ and } x_{k-1}] = \frac{1}{m_k p_s} v_{q_k, n-k}(x_{k-1} + t_k) = \frac{m_{k+1}}{m_k p_s}$$

Plugging this in (20) we get the expected result (19).

- When b = 0 (and thus  $w_k = T$ ), a similar reasoning permits to prove (19) which is then true in both cases.

## **B** Preprocessing of Algorithm 2.

#### Algorithm 3 Preprocessing for Algorithm 2

1: for  $p \in Q$  do 2: define  $v_{p,0}(x) = 1_{p \in F}$ . 3: for k = 1 to n do 4: compute  $v_{p,k}(x)$  using (17). 5: end for 6: end for

**Proposition 4.** Algorithm 3 has space and time complexity  $O(|Q|n^2)$ . Its bit space complexity is  $O(|Q|n^3)$ .

*Proof (Sketch).* The polynomial  $v_{q,m}$  is of degree m, it has O(m) coefficients. Therefore the time and space complexity are  $O(\sum_{m=1}^{n} |Q|m) = O(|Q|n^2)$ . Magnitudes of coefficients of  $v_{q,m}$  behave like  $2^{m\mathcal{H}}$  where  $\mathcal{H}$  is the entropy of

Magnitudes of coefficients of  $v_{q,m}$  behave like  $2^{m\mathcal{H}}$  where  $\mathcal{H}$  is the entropy of the timed language (see [3]) and thus one needs O(m) bits to store them. This explains why an extra factor n appears when dealing with bit space complexity.

<sup>&</sup>lt;sup>6</sup> Its inverse (t function of r) is known as the quantile function.

## C Examples

In section C.1 we show how Algorithm 1 applies to the classical example of alternating permutations. In section C.2 we apply this algorithm to the running example of up-up-down-down permutations. In section C.3 we treat the example of permutations without two consecutive descents.

#### C.1 The alternating permutations



**Fig. 2.** An automaton for  $(\mathbf{da})^*(\varepsilon + \mathbf{d})$  and its S-T encoding of type  $\mathbf{d}$ 

The class of alternating permutation is<sup>7</sup>  $Alt = \mathfrak{S}_0 \cup sg^{-1}[(da)^*(\varepsilon + d)]$ . It is well known since the 19<sup>th</sup> century and the work of Désiré André that

$$EGF(Alt)(z) = \tan(z) + \sec(z)$$
 (where  $\sec(z) = 1/\cos(z)$ ).

Several different proofs of this results can be found in [16]. Here we give a novel proof based on the application of Algorithm 1 on  $(\mathbf{da})^*(\varepsilon + \mathbf{d})$ .

A prolongation of  $(\mathbf{da})^*(\varepsilon + \mathbf{d})$  is  $(\mathbf{da})^*(\mathbf{d} + \mathbf{da})$ . We add  $\varepsilon$  to the language to add 1 to its VGF, indeed

$$EGF(\texttt{Alt})(z) = 1 + VGF[(\mathbf{da})^*(\mathbf{d} + \mathbf{da})](z) = VGF[(\mathbf{da})^*(\varepsilon + \mathbf{d})](z)$$

The S-T encoding of type **a** of  $(\mathbf{da})^*(\varepsilon + \mathbf{d})$  is just S<sup>\*</sup> which is recognized by the one loop automaton depicted in the right of Figure 2. Thus  $M_{\mathsf{S}} = (1), M_{\mathsf{T}} = (0)$  and we must compute  $\exp(zM) = \sum_{n \in \mathbb{N}} z^n M^n / n!$  with  $M = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ .

Computation of  $\exp(zM)$  is easy since M is unipotent and thus its sequence of power  $M^k$  is periodic:  $M^0 = I_2$ ,  $M^1 = M$ ,  $M^2 = -I_2$ ,  $M^3 = -M$ ,  $M^3 = I_2$ ,  $M^4 = M$ , ...

Then for all  $k \ge 0$ :

$$M^{2k} = \begin{pmatrix} (-1)^k & 0\\ 0 & (-1)^k \end{pmatrix}; \quad M^{2k+1} = \begin{pmatrix} 0 & (-1)^{2k}\\ (-1)^{2k+1} & 0 \end{pmatrix}$$

Hence  $\exp(zM) = \sum_{n \in \mathbb{N}} z^n M^n / n! = \begin{pmatrix} \cos(z) - \sin(z) \\ \sin(z) & \cos(z) \end{pmatrix}$ .

<sup>&</sup>lt;sup>7</sup> The unique permutation on the empty set has no signature and thus  $\mathfrak{S}_0 \not\subseteq \mathbf{sg}^{-1}(L)$  for any language L of signature.

By definition  $A_1(z) = \cos(z), A_2(z) = -\sin(z)$ . We can conclude:

$$EGF(Alt)(z) = A_1(z)^{-1}(1 - A_2(z)) = \frac{1}{\cos(z)} + \tan(z).$$

#### C.2 The up-up-down-down permutations

Here we compute the exponential generating function of the class of up-updown-down permutations given as running example of the article. Recall that the corresponding regular language is  $L^{ex} = (\mathbf{aadd})^*(\mathbf{aa} + \varepsilon)$ , one of its extension is  $L^{ex'} = (\mathbf{aadd})^*(\mathbf{aad} + \mathbf{a})$  and the S-T-encoding of type **d** of this latter language is  $\mathbf{st}_{\mathbf{d}}(L') = (\mathbf{TS})^*\mathbf{T}$ . These languages are recognized by automata depicted in Figure 1. The adjacency matrices of the third automaton are  $M_{\mathbf{S}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ ,  $M_{\mathbf{T}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$  and the row vector of final state is  $\mathbf{F} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ . Let  $M = \begin{pmatrix} -M_{\mathbf{S}} - M_{\mathbf{T}} \\ M_{\mathbf{T}} & M_{\mathbf{S}} \end{pmatrix}$ . Again the computation of  $\exp(zM)$  is easy since M is unipotent<sup>8</sup>:

$$M = \begin{pmatrix} 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}; M^{2} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}; M^{3} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix}; M^{4} = I_{4}.$$

Thus if we denote by  $f_i(z) = \sum_{n=0}^{+\infty} z^{4n+i}/(4n+i)!$  for  $i \in \{0, 1, 2, 3\}$  we have:

$$\exp zM = f_0(z)I + f_1(z)M + f_2(z)M^2 + f_3(z)M^3 \text{ and}$$
$$A_1(z) = \begin{pmatrix} f_0(z) & -f_3(z) \\ -f_1(z) & f_0(z) \end{pmatrix}; A_2(z) = \begin{pmatrix} f_2(z) & -f_1(z) \\ f_3(z) & f_2(z) \end{pmatrix}.$$

The function  $f_i$  can be expressed with trigonometric and hyperbolic functions:

$$f_0(z) = [\cosh(z) + \cos(z)]/2; \quad f_1(z) = [\sinh(z) + \sin(z)]/2; f_2(z) = [\cosh(z) - \cos(z)]/2; \quad f_3(z) = [\sinh(z) - \sin(z)]/2.$$

We have

$$[I_2 - A_2(z)]\mathbf{F} = \begin{pmatrix} f_1(z) \\ 1 - f_2(z) \end{pmatrix}$$

and thus

$$\begin{pmatrix} f_p(z) \\ f_q(z) \end{pmatrix} = \begin{pmatrix} f_0(z) & -f_3(z) \\ -f_1(z) & f_0(z) \end{pmatrix}^{-1} \begin{pmatrix} f_1(z) \\ 1 - f_2(z) \end{pmatrix}.$$

Using Cramer formula we get  $f_p(z) = [f_1(z)f_0(z) + f_3(z)(1 - f_2(z))]/[f_0^2(z) + f_1(z)f_3(z)]$ . After straightforward simplifications we obtain the wanted result:

$$f(z) = f_p(z) = \frac{\sinh(z) - \sin(z) + \sin(z)\cosh(z) + \sinh(z)\cos(z)}{1 + \cos(z)\cosh(z)}$$



**Fig. 3.** From left to right automata for  $L^{ex_3}$ ,  $L^{ex'_3} = \{\varepsilon\} \cup L^{ex_3}$ .  $\{\mathbf{a}\}$  and  $\mathtt{st}_{\mathbf{a}}(L^{ex'_3})$ 

#### C.3 Permutations without two consecutive descents

Consider the class  $C^{ex_3}$  of permutations without two consecutive descents. This class has already been studied and its EGF computed. References and many details can be found in the On-Line Encyclopedia of Integer Sequences (OEIS), sequence A049774. In particular the following EGF is given:

$$EGF(C^{ex_3})(z) = \frac{\sqrt{3}e^{z/2}}{\sqrt{3}\cos\left(\frac{\sqrt{3}}{2}z\right) - \sin\left(\frac{\sqrt{3}}{2}z\right)}$$

We give an alternative proof of this result based on the method developed in this article. The class  $C^{ex_3}$  can be described in terms of regular languages:

$$C^{ex_3} = \mathfrak{S}_0 \cup \mathbf{sg}^{-1}[(\mathbf{a} + \mathbf{da})^*(\varepsilon + \mathbf{d})].$$

A prolongation of  $(\mathbf{a} + \mathbf{da})^* (\varepsilon + \mathbf{d})$  is  $(\mathbf{a} + \mathbf{da})^* \mathbf{a}$ . As for alternating permutations we add the word  $\varepsilon$  to this language to add 1 to the final generating function, thus we get the language  $(\mathbf{a} + \mathbf{da})^*$  recognized by the automaton depicted in the middle of Figure 3. Its S-T encoding of type  $\mathbf{a}$  is  $(\mathbf{S} + \mathbf{TT})^*$  which is recognized by the automaton depicted in the right of Figure 3. Its adjacency matrices are  $M_{\mathbf{S}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ ,  $M_{\mathbf{T}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$  and the row vector of final state is  $\mathbf{F} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ . Let  $M = \begin{pmatrix} -M_{\mathbf{S}} - M_{\mathbf{T}} \\ M_{\mathbf{T}} & M_{\mathbf{S}} \end{pmatrix}$ . We will solve directly the differential equation (9) with

boundary condition (10), i.e. the system

$$\frac{\partial f_p}{\partial x}(x,z) = -zf_p(x,z)dy - zf_q(1-x,z)dy;$$
(21)

$$\frac{\partial f_q}{\partial x}(x,z) = -zf_p(1-x,z).$$
(22)

with boundary conditions  $f_p(1,z) = 1$ ;  $f_q(1,z) = 0$  Equation (21) taken at x = 1 ensures that  $\frac{\partial f_p}{\partial x}(1,z) = -zf_p(0,z) - zf_q(1,z) = -zf_p(0,z)$ . Thus we have the boundary conditions

$$f_p(1,z) = 1;$$
 (23)

$$\frac{\partial f_p}{\partial x}(1,z) = -zf_p(0,z). \tag{24}$$

<sup>&</sup>lt;sup>8</sup> This is in fact the case for all cyclic automata.

Differentiating (21) and replacing  $\frac{\partial f_q}{\partial x}(1-x,z)$  using (22) we get:

$$\frac{\partial^2 f_p}{\partial x^2}(x,z) = -z \frac{\partial f_p}{\partial x} - z^2 f_p(x,z);$$
(25)

Solutions are of the form:  $f_p(x, z) = e^{-zx/2} \left[ a(z) \cos\left(\frac{\sqrt{3}}{2}zx\right) + b(z) \sin\left(\frac{\sqrt{3}}{2}zx\right) \right]$ with a(z) and b(z) to be determined using boundary conditions (23) and (24) i.e. a(z) and b(z) should satisfy:

$$\cos\left(\frac{\sqrt{3}}{2}z\right)a(z) + \sin\left(\frac{\sqrt{3}}{2}z\right)b(z) = e^{z/2};$$
$$a(z) + \sqrt{3} \qquad b(z) = 0.$$

Solving this system we obtained the expected EGF:

$$EGF(C^{ex_3})(z) = f_p(0, z) = a(z) = \frac{\sqrt{3}e^{z/2}}{\sqrt{3}\cos\left(\frac{\sqrt{3}}{2}z\right) - \sin\left(\frac{\sqrt{3}}{2}z\right)}$$