Counting and generating permutations using timed languages (long version)

Nicolas Basset

Counting and generating permutations using timed languages (long version) ⋆ ⋆⋆

Nicolas Basset[1,2]

[1] LIGM, University Paris-Est Marne-la-Vallée and CNRS, France.
[2] LIAFA, University Paris Diderot and CNRS, France
nbasset@liafa.univ-paris-diderot.fr

Abstract. The signature of a permutation σ is a word $\mathtt{p}(\sigma) \subseteq \{U, D\}^*$ whose i^{th} letter is U when σ goes "up" (i.e. $\sigma(i) > \sigma(i+1)$) and is D when σ goes "down" (i.e. $\sigma(i) < \sigma(i+1)$). Combinatorics of permutations with a prescribed signature is a quite well explored topics. Languages of signatures permit to express a broad number of classes of permutations (e.g. the permutations without two consecutive downs). Here we state and address the two problems of counting and randomly generating in the set $\mathtt{p}^{-1}(L)$ of permutations whose signature is in a given regular language $L \subseteq \{U, D\}^*$. First we give an algorithm that computes a closed form formula for the exponential generating function of $\mathtt{p}^{-1}(L)$. Then we give an algorithm that generates randomly the n-length permutations of $\mathtt{p}^{-1}(L)$ in a uniform manner i.e. all the permutations of a given length with a signature in L are equally probable to be returned. Both contributions are based on a geometric interpretation of a subclass of timed regular languages.

1 Introduction

The signature of a permutation $\sigma = \sigma_1 \cdots \sigma_n$ is the word $w = w_1 \cdots w_{n-1} \in \{U, D\}^{n-1}$ with $w_i = D$ when σ goes "down" ($\sigma_i > \sigma_{i+1}$), and $w_i = U$ when it goes "up" ($\sigma_i < \sigma_{i+1}$).

Generating all the permutations with a prescribed signature or simply count them are two classical combinatorial topics (see e.g. [11] and reference therein).

A very well studied example of permutations given by their signatures are the so-called alternating (or zig-zag, or down-up) permutations (see [9] for a survey). Their signatures belong to the language expressed by the regular expression $(DU)^*(D + \epsilon)$ (i.e. they satisfy $\sigma_1 > \sigma_2 < \sigma_3 > \sigma_4 ...$).

Such a definition of class of permutations in terms of a language of signatures is in fact a novelty of the present paper. To a language $L \subseteq \{U, D\}^*$, we associate the class $\mathtt{p}^{-1}(L)$ of permutations whose signature is in L. Many classes of permutations can be expressed in that way (e.g. alternating permutations, those without 2 consecutive downs, those with an even number of downs, etc.).

⋆ The present paper is a long version of an article submitted to MFCS 2013.

⋆⋆ The support of Agence Nationale de la Recherche under the project EQINOCS (ANR-11-BS02-004) is gratefully acknowledged.

We state and address the two problems of counting and randomly generating when the language of signatures is regular.

We propose Algorithm 1 which takes as its input a regular language L and returns a closed form formula for the exponential generating function (EGF) of $\mathtt{p}^{-1}(L)$ i.e. 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 [7] for an overview of analytic combinatorics).

The random generation is done by an algorithm described in Theorem 4. The regular language of signatures L together with n the size of permutation to generate are given in input and the output are random permutations of size n whose signatures are in L with equal probability to be returned.

Our theory is based on a geometric interpretation of timed regular languages initiated in [3]. In that paper the authors introduce the concept of volume and entropy of timed regular languages as well as recurrent equations on timed languages and their volume. With these authors, we have defined and characterize volume generating function of timed language in [2]. In this latter paper a link between enumerative combinatorics and timed regular languages was foreseen. Here we establish such a link. In fact 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 [10]. 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.

Paper structure. In section 2 we expose the problem statements. In section 3 we establish the link between the classes of permutations associated with languages of signatures and timed languages of a particular form. We address the two problems in section 4 and discuss our results and perspectives in the last section. Proofs and detailed examples are given in the appendix.

Submitted version. The present paper is a long version of an article submitted to MFCS 2013.

2 Two problem statements

All along the paper we use the two letter alphabet $\{U, D\}$ whose elements must be read as "up" and "down". Words of $\{U, D\}^*$ are called signatures. For $n \in \mathbb{N}$ we denote $[n] = \{1, \ldots, n\}$ and by $\mathfrak{S}_n$ the set of permutation of $[n]$. We also use the one line notation of permutations e.g. $\sigma = 231$ 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 $u = u_1 \cdots u_{n-1} \in \{U, D\}^{n-1}$ denoted by $\mathtt{p}(\sigma)$ such that for $i \in [n]$, $\sigma_i < \sigma_{i+1}$ iff $u_i = U$ (we speak of an "up", also known as an ascent) and $\sigma_i > \sigma_{i+1}$ iff $u_i = D$ (we speak of a "down", also known as a descent) e.g. $\mathtt{p}(21354) = \mathtt{p}(32451) = DUUD$.

This statistics appears in the literature under several different names and forms such as descent set, ribbon diagram, etc. The usual definition of signature of a permutation is an n-tuple of $+1$ ("up") and -1 ("down"). Here we use words to express in a very convenient way constraints on permutations in terms of languages. More precisely we are interested in $\mathtt{p}^{-1}(L) = \{\sigma \mid p(\sigma) \in L\}$: the class of permutations with a signature in $L \subseteq \{U, D\}^*$. Given a language L we denote by L_n the sub-language of L restricted to words of length n. The exponential generating function of $\mathtt{p}^{-1}(L)$ is

$$EGF[\mathtt{p}^{-1}(L)](z) = \sum_{\sigma \in \mathtt{p}^{-1}(L)} \frac{z^{|\sigma|}}{|\sigma|!} = \sum_{n \geq 1} |\mathtt{p}^{-1}(L_{n-1})| \frac{z^n}{n!} = \sum_{u \in L} |\mathtt{p}^{-1}(u)| \frac{z^{|u|+1}}{(|u|+1)!}.$$

Example 1. Consider as a running example the class C^{ex} of permutation without two consecutive downs. Then[3] $C^{ex} = \mathfrak{S}_0 \cup \mathtt{p}^{-1}(L^{ex})$ where L^{ex} is the language expressed by the regular expression $(U + DU)^*(D + \epsilon)$. The theory developed in the paper[4] permits to find the exponential generating function of C^{ex}:

$$EGF(C^{ex})(z) = \frac{3\cos(z\sqrt{3}/2) + \sqrt{3}\sin(z\sqrt{3}/2)}{[2\cos(z\sqrt{3}/2) - 1][2\cos(z\sqrt{3}/2) + 1]} e^{z/2}$$

The following Taylor expansion gives the cardinalities of sets of n-length permutations of C^{ex} for $n \in \mathbb{N}$ e.g. there are 5 permutations of length 3 in C^{ex}.

$$EGF(C^{ex})(z) = 1 + z + 2\frac{z^2}{2!} + 5\frac{z^3}{3!} + 17\frac{z^4}{4!} + 70\frac{z^5}{5!} + 349\frac{z^6}{6!} + 2017\frac{z^7}{7!} + \dots$$

Now we state the two problems solved in the paper.

Problem 1. Design an algorithm which takes as input a regular language $L \subseteq \{U, D\}^*$ and returns a closed form formula for $EGF(\mathtt{p}^{-1}(L))$

Problem 2. Design an algorithm which takes as input a regular language $L \subseteq \{U, D\}^*$ and a positive integer n and returns a random permutation σ uniformly in $\mathtt{p}^{-1}(L_{n-1})$ i.e. such that the probability for each $\sigma \in \mathtt{p}^{-1}(L_{n-1})$ to be returned is $1/|\mathtt{p}^{-1}(L_{n-1})|$.

3 A timed and geometric approach

In section 3.1 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 the sequence $(\mathcal{O}_n(L))_{n \geq 1}$ and generating points uniformly in $\mathcal{O}_n(L)$. Then we define timed languages $\mathbb{L}'$ associated to L as well as its volume (section 3.2) and describe a volume preserving transformation between $\mathcal{O}_n(L)$ and $\mathbb{L}'_n$.

[3] The unique permutation on the empty set has no signature and thus $\mathfrak{S}_0 \not\subseteq \mathtt{p}^{-1}(L)$ for any language L of signature.

[4] Detailed examples are given in the appendix.

3.1 Order sets of a language of signatures $(\mathcal{O}_n(L))_{n\geq 1}$

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

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 [10][5] of σ and denoted by $\mathcal{O}(\sigma)$ e.g. $\boldsymbol{\nu} = (0.3, 0.2, 0.4, 0.5, 0.1)$ 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 σ from $\boldsymbol{\nu}$ it suffices to use a sorting algorithm. In this optic, we will denote by $\mathtt{sort}(\boldsymbol{\nu})$ the permutation σ returned by the sorting algorithm on $\boldsymbol{\nu}$ i.e. such that $0 \leq \nu_{\sigma_1^{-1}} \leq \ldots \leq \nu_{\sigma_n^{-1}} \leq 1$.

The *signature* of a vector $\boldsymbol{\nu} \in [0,1]^n$ is the word $\mathtt{sg}(\boldsymbol{\nu}) = \mathtt{p}(\mathtt{sort}(\nu))$ i.e. such that $\nu_i < \nu_{i+1}$ iff $u_i = U$ and $\nu_i > \nu_{i+1}$ iff $u_i = D$. e.g. $\mathtt{p}(0.3, 0.2, 0.4, 0.5, 0.1) = DUUD$. The *order polytope* [10] of a signature $u \in \{U, D\}^{n-1}$ is the polytope $\mathcal{O}(u) = \{\boldsymbol{\nu} \in [0,1]^n \mid \mathtt{sg}(\boldsymbol{\nu}) = u\}$. It is clear that the collection of order simplices $\mathcal{O}(\sigma)$ with all σ having the same signature u form an almost disjoint partition of the order polytope $\mathcal{O}(u)$: $\mathcal{O}(u) = \bigsqcup_{\sigma \in \mathtt{p}^{-1}(u)} \mathcal{O}(\sigma)$ (e.g. $\mathcal{O}(DUU) = \mathcal{O}(2134) \sqcup \mathcal{O}(3124) \sqcup \mathcal{O}(4123)$). Passing to volume we get:

$$\mathtt{Vol}(\mathcal{O}(u)) = \sum_{\sigma \in \mathtt{p}^{-1}(u)} \mathtt{Vol}(\mathcal{O}(\sigma)) = \frac{|\mathtt{p}^{-1}(u)|}{n!} \tag{1}$$

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 $\mathcal{O}_n(L)$:

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

For volumes we get:

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

Reformulating the two problems with the geometric approach As a consequence of 3, Problem 1 can be reformulated as computing the *volume generating function* (VGF) of the sequence $\mathcal{O}(L) =_{def} (\mathcal{O}_n(L))_{n\geq 1}$:

$$VGF(\mathcal{O}(L))(z) =_{def} \sum_{n \geq 1} \mathtt{Vol}(\mathcal{O}_n(L)) z^n = EGF(\mathtt{p}^{-1}(L))(z) \tag{4}$$

[5] Order simplices, order and chain polytopes of signatures defined here are particular cases of Stanley's order and chain polytopes.

Problem 2 can also be treated using order polytopes $\mathcal{O}_n(L)$. Indeed it suffices to generate uniformly a vector $\nu \in \mathcal{O}_n(L)$ and then sort it to get a permutation $\sigma = \mathtt{sort}(\nu)$. As the simplices $\mathcal{O}(\sigma)$ for $\sigma \in \mathtt{p}^{-1}(L_n)$ 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 ν, and thus all $\sigma \in \mathtt{p}^{-1}(L_n)$ have the same probability to be chosen.

We have seen with (2) that permutations of a fixed length n fits well with the n^{th} order set. However, it is not clear how to fit the sequence of order sets (when n varies) with the dynamics of the language L. It is easier to handle a timed language $\mathbb{L}$ since its sequence of volume $(\mathtt{Vol}(\mathbb{L}_n))_{n\in\mathbb{N}}$ satisfy a recursive equation [3,2]. We will find a volume preserving transformation between order sets $\mathcal{O}(n)$ and timed languages $(\mathbb{L}_n)_{n\in\mathbb{N}}$ and hence reduce Problem 1 to the computing of the ordinary generating function of $(\mathtt{Vol}(\mathbb{L}_n))_{n\in\mathbb{N}}$. For the second problem, by generating uniformly a timed word in $\mathbb{L}_n$ and applying the volume preserving transformation we will get a uniform random point in $\mathcal{O}_n(L)$.

3.2 Timed semantic of a language of signatures: $\left(\mathbb{L}'_{\boldsymbol{n}}\right)_{\boldsymbol{n}\in\mathbb{N}}$

This section is inspired by timed automata theory and designed for non experts. We adopt a non standard[6] 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 Σ 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 $(t_1, \ldots, t_n) \in \mathbb{R}^n$ together with a word of events $w = w_1 \cdots w_n \in \Sigma^n$. We adopt the following convention, we write $(\boldsymbol{t}, w)$ the timed word $(t_1, w_1) \cdots (t_n, w_n)$ with $\boldsymbol{t} = (t_1, \cdots, t_n)$ and $w \in \Sigma^n$ $(n \geq 1)$. Continuing with the same convention, given a timed language $\mathbb{L} \subseteq (\mathbb{R}^+ \times \Sigma)^*$, then the timed language restricted to words of length n, A_n can be seen as a formal union of sets $\biguplus_{w\in\Sigma^n} \mathbb{L}'_w \times \{w\}$ where $\mathbb{L}'_w = \{\boldsymbol{t} \in \mathbb{R}^n \mid (\boldsymbol{t}, w) \in \mathbb{L}\}$ 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 a $\mathbb{L}'_n$ one can associate a sequence of volumes and a VGF as follows:

$$\mathtt{Vol}(\mathbb{L}'_n) = \sum_{w\in\Sigma^n} \mathtt{Vol}(A_w); \quad VGF(\mathbb{L})(z) = \sum_{w\in\Sigma^*} \mathtt{Vol}(A_w) z^{|w|} = \sum_{n\in\mathbb{N}} \mathtt{Vol}(\mathbb{L}'_n) z^n$$

The clock semantic of a signature. A clock is a non-negative real variable. Here we only consider two *clocks* bounded by 1 and denoted by x^U and x^D. A clock word is a tuple whose component are a starting clock vector $(x_0^U, x_0^D) \in$

[6] We refer the reader to [1] for a standard approach of timed automata theory.

$[0,1]$, a timed word $(t_1,a_1)\cdots(t_n,a_n) \in ([0,1]\times\{U,D\})^*$ and an ending clock vector $(x_n^U,x_n^D)\in[0,1]^2$, it is denoted by $(x_0^U,x_0^D) \xrightarrow{(t_1,a_1)\cdots(t_n,a_n)} (x_n^U,x_n^D)$.

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 $(\mathbf{x}_0 \xrightarrow{\mathbf{w}} \mathbf{x}_1)\cdot(\mathbf{x}_2 \xrightarrow{\mathbf{w}'} \mathbf{x}_3) = \mathbf{x}_0 \xrightarrow{\mathbf{ww}'} \mathbf{x}_3$. A *clock language* is a set of clock words. The product of two clock languages $\mathcal{L}$ and $\mathcal{L}'$ is

$$\mathcal{L}\cdot\mathcal{L}' = \{c\cdot c' \mid c\in\mathcal{L},\ c'\in\mathcal{L}',\ c \text{ and } c' \text{ compatible}\}. \tag{5}$$

The clock languages[7] $\mathcal{L}(U)$ (resp. $\mathcal{L}(D)$) associated to an up (resp. a down) is the set of clock words of the form $(x^U,x^D)\xrightarrow{(t,U)}(x^U+t,0)$ (resp. $(x^U,x^D)\xrightarrow{(t,D)}(0,x^D+t)$) and such that $x^U+t\in[0,1], x^D+t\in[0,1]$ (and by definition of clocks and delays $x^U\geq 0, x^D\geq 0, t\geq 0$). These definitions extends inductively to all signatures $\mathcal{L}(u_1\cdots u_n)=\mathcal{L}(u_1)\cdots\mathcal{L}(u_n)$ (using the product of clock languages as defined in 5).

Example 2. $(0,0)\xrightarrow{(0.7,D)(0.2,U)(0.2,U)(0.5,D)}(0,0.5)\in\mathcal{L}(DUUD)$ since
$(0,0)\xrightarrow{(0.7,D)}(0,0.7)\in\mathcal{L}(D)$; $(0,0.7)\xrightarrow{(0.2,U)}(0.2,0)\in\mathcal{L}(U)$;
$(0.2,0)\xrightarrow{(0.2,U)}(0.4,0)\in\mathcal{L}(U)$; $(0.4,0)\xrightarrow{(0.5,U)}(0,0.5)\in\mathcal{L}(D)$.

The timed semantic of a language of signatures. The *timed polytope* associated to a signature $w\in\{U,D\}^*$ is $P_w =_{def} \{\boldsymbol{t} \mid (0,0)\xrightarrow{(\boldsymbol{t},w)}\mathbf{y}\in\mathcal{L}(w) \text{ for some } \mathbf{y}\in[0,1]^2\}$ e.g. $(0.7,0.2,0.2,0.5,0.1)\in P_{DUUDU}$. The definition of such a timed polytope will be clarified in proposition 1 and its following example. The timed semantic of a language of signatures L' is

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

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

The *chain polytope* [10] of a signature u is the set $\mathcal{C}(u)$ of vectors $\boldsymbol{t}\in[0,1]^n$ such that for all $i<j\leq n$ and $l\in\{U,D\}$, $w_i\cdots w_{j-1}=l^{j-i}\Rightarrow t_i+\ldots+t_j\leq 1$.

Proposition 1. *Given a word $u\in\{U,D\}^*$ and $l\in\{U,D\}$ then the timed polytope of ul is the chain polytope of u: $P_{ul}=\mathcal{C}(u)$.*

Example 3. A vector $(t_1,t_2,t_3,t_4,t_5)\in[0,1]^5$ belongs to $P_{DUUDU}=\mathcal{C}(DUUD)$ 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 $(1-t_1,t_2,t_2+t_3,1-t_4,t_5)\in\mathcal{O}(DUUD)$. One can check this fact on examples given before: $(0.7,0.2,0.2,0.5,0.1)\in P_{DUUDU}$ corresponds to the vector $(0.3,0.2,0.4,0.5,0.1)\in\mathcal{O}(DUUD)$.

[7] A reader acquainted with timed automata would have noticed that the clock language $\mathcal{L}(U)$ (resp. $\mathcal{L}(D)$) corresponds to a transition of a timed automaton where the guards $x^U\leq 1$ and $x^D\leq 1$ are satisfied and where x^D (resp. x^U is reset).

The purpose of the following section is to give the general formula for the correspondence between timed polytopes and order polytopes we have foreseen in the previous example.

3.3 Volume preserving transformation between $\mathbb{L}'_n$ and $\mathcal{O}_n(L)$.

Let n be a positive integer. We define for $w = ul$ with $u \in \{U, D\}^{n-1}$ and $l \in \{U, D\}$ a volume preserving function $(t_1, \cdots, t_n) \mapsto (\nu_1, \cdots, \nu_n)$ the chain polytope $\mathcal{C}(u) = P_{ul}$ to the order polytope $\mathcal{O}(u)$. This is a simple case of Theorem 2.1 of [8].

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

Proposition 2. *The mapping $\phi_{ul} : (t_1, \cdots, t_n) \mapsto (\nu_1, \cdots, \nu_n)$ is a volume preserving transformation from $\mathcal{C}(u) = P_{ul}$ to $\mathcal{O}(u)$. It can be computed in linear time using the following recursive characterization:* $\begin{vmatrix} \nu_1 = t_1 & \text{if } w_1 = U \\ \nu_1 = 1 - t_1 & \text{if } w_1 = D \end{vmatrix}$ *and for $i \geq 2$:* $\begin{vmatrix} \nu_i = \nu_{i-1} + t_i & \text{if } w_{i-1}w_i = UU; \\ \nu_i = t_i & \text{if } w_{i-1}w_i = DU; \\ \nu_i = 1 - t_i & \text{if } w_{i-1}w_i = UD; \\ \nu_i = \nu_{i-1} - t_i & \text{if } w_{i-1}w_i = DD. \end{vmatrix}$

Proposition 2 links the timed polytope of a signature of length $n+1$ and the order polytopes of a signature of length n. We correct this mismatch of length using prolongation of languages. We say that a language L' is 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 L has prolongations e.g. $L' = Ll$ for $l \in \{U, D\}$ are prolongations of L.

Now we can extend proposition 2 to a language of signatures.

Theorem 1. *Let $L \subseteq \{U, D\}^*$ and $\mathbb{L}'$ be the timed semantic 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.*

$$\begin{array}{rl} \phi : \ \mathbb{L}'_n & \to \mathcal{O}_n(L) \\ (\boldsymbol{t}, w) & \mapsto \ \phi_w(\boldsymbol{t}) \end{array} \tag{6}$$

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 our previous work [2]. Nevertheless the equations of this article were quite uneasy to handle and did not give a closed form formula for the VGF. To get more precise and simple equations than in [2] we work with a novel class of timed languages involving two kinds of transitions S and T.

3.4 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*. A word $w'_1\cdots w'_n \in \{S,T\}^*$ is the S-T-encoding of type $l \in \{U,D\}$ of a word $w_1\cdots w_n \in \{U,D\}^*$ whenever for every $i \in [n]$, $w'_i = S$ if $w_i = w_{i-1}$ and $w'_i = T$ otherwise, with convention that $w_0 = l$. We denote by $\mathtt{st}_l(w)$ the S-T-encoding of a word $w \in \{U,D\}^*$. The mapping $\mathtt{st}_l$ is invertible with inverse defined by $\mathtt{st}_l^{-1}(w'_1\cdots w'_n) = w_1\cdots w_n$ where for every $i \in [n]$, $w_i = w_{i-1}$ if $w'_i = S$ and $w_i \neq w_{i-1}$ otherwise, with convention that $w_0 = l$. Notion of S-T-encodings can be extended naturally to languages. e.g. $\mathtt{st}_U[(U+DU)^*] = (S+TT)^*$. We call an S-T-automaton, a deterministic finite state automaton with transition alphabet $\{S,T\}$.

Timed semantic and *S-T*-encoding In the following we define clock and timed languages similarly to what we have done in section 3.2. Here we need only one clock x that remains bounded by 1. We define the clock languages associated to S by $\mathcal{L}(S) = \{x \xrightarrow{(t,S)} x+t \mid x \in [0,1], t \in [0,1-x]\}$ and the clock language associated to T by $\mathcal{L}(T) = \{x \xrightarrow{(t,T)} t \mid x \in [0,1], t \in [0,1-x]\}$. Let $L'' \subseteq \{S,T\}^*$ we denote by $L''(x)$ the timed language starting from x: $L''(x) = \{(\boldsymbol{t},w) \mid \exists y \in [0,1], x \xrightarrow{(\boldsymbol{t},w)} y \in \mathcal{L}(w), w \in L''\}$. The *timed semantic* of $L'' \subseteq \{S,T\}^*$ is $L''(0)$.

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

Proposition 3. *Let $L' \subseteq \{U,D\}^*$, $l \in \{U,D\}$, $\mathbb{L}'$ be the timed semantic of L' and $\mathbb{L}''$ be the timed semantic of $\mathtt{st}_l(L')$ then the function $(\boldsymbol{t},w) \mapsto (\boldsymbol{t}, \mathtt{st}_l^{-1}(w))$ is a volume preserving transformation from $\mathbb{L}''_n$ to $\mathbb{L}'_n$.*

Using notation and results of Theorem 1 and Proposition 3 we get a volume preserving transformation from $\mathbb{L}''_n$ to $\mathcal{O}_n(L)$.

Theorem 2. *The function $(\boldsymbol{t},w) \mapsto \phi_{\mathtt{st}_l^{-1}(w)}(\boldsymbol{t})$ is a volume preserving transformation from $\mathbb{L}''_n$ to $\mathcal{O}_n(L)$ computable in linear time. In particular*

$$\mathtt{Vol}(\mathbb{L}''_n) = \frac{|\mathfrak{p}^{-1}(L_{n-1})|}{n!} \text{ for } n \geq 1, \text{ and } VGF(\mathbb{L}'')(z) = EGF(\mathfrak{p}^{-1}(L))(z)$$

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

4 Solving the two problems

4.1 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 solve Problem 1.

We have defined just above timed language $L''(x)$ parametrized by an initial clock vector x. Given an S-T-automaton, we can also consider the intial

state as a parameter and write Kleene like systems of equations on parametric language $L_p(x)$ (similarly to [2]). More precisely, let $\mathcal{A} = \{\{S,T\}, Q, i, F, \delta\}$ be S-T-automaton. To every state $p \in Q$ we denote by $L_p \subseteq \{S,T\}^*$ the language starting from p i.e. recognized by $\mathcal{A}_p =_{def} \{\{S,T\}, Q, p, F, \delta\}$. We adopt the convention that $L_{\delta(p,l)}$ is empty when $\delta(p,l)$ is undefined and the corresponding generating function is null. Then for every $p \in Q$, we have a parametric language equation:

$$L_p(x) = \left[\cup_{t \leq 1-x}(t,S)L_{\delta(p,S)}(x+t)\right] \cup \left[\cup_{t \leq 1-x}(t,T)L_{\delta(p,T)}(t)\right] \cup (\epsilon \text{ if } p \in F) \tag{7}$$

Passing to volume generating functions $f_p(x,z) =_{def} VGF(L_p(x))(z)$ (as in [2]) we get:

$$f_p(x,z) = z\int_x^1 f_{\delta(p,S)}(s,z)ds + z\int_0^{1-x} f_{\delta(p,T)}(t,z)dt + (1 \text{ if } p \in F) \tag{8}$$

In matrix notation:

$$\boldsymbol{f}(x,z) = zM_S\int_x^1 \boldsymbol{f}(s,z)ds + zM_T\int_0^{1-x} \boldsymbol{f}(t,z)dt + \boldsymbol{F} \tag{9}$$

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

Equation (9) is equivalent to the differential equation:

$$\frac{\partial}{\partial x}\boldsymbol{f}(x,z) = -zM_S\boldsymbol{f}(x,z) - zM_T\boldsymbol{f}(1-x,z) \tag{10}$$

with boundary condition

$$\boldsymbol{f}(1,z) = \boldsymbol{F}. \tag{11}$$

Equation (10) is equivalent to the following linear homogeneous system of ordinary differential equations with constant coefficients:

$$\frac{\partial}{\partial x}\begin{pmatrix}\boldsymbol{f}(x,z)\\ \boldsymbol{f}(1-x,z)\end{pmatrix} = z\begin{pmatrix}-M_S & -M_T\\ M_T & M_S\end{pmatrix}\begin{pmatrix}\boldsymbol{f}(x,z)\\ \boldsymbol{f}(1-x,z)\end{pmatrix}. \tag{12}$$

whose solution is of the form

$$\begin{pmatrix}\boldsymbol{f}(x,z)\\ \boldsymbol{f}(1-x,z)\end{pmatrix} = \exp\left[xz\begin{pmatrix}-M_S & -M_T\\ M_T & M_S\end{pmatrix}\right]\begin{pmatrix}\boldsymbol{f}(0,z)\\ \boldsymbol{f}(1,z)\end{pmatrix} \tag{13}$$

Taking $x = 1$ in (13) and using the boundary condition (11) we obtain:

$$\begin{aligned}\boldsymbol{F} &= A_1(z)\boldsymbol{f}(0,z) + A_2(z)\boldsymbol{F}\\ \boldsymbol{f}(0,z) &= A_3(z)\boldsymbol{f}(0,z) + A_4(z)\boldsymbol{F}\end{aligned} \tag{14}$$

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_T and M_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 $\boldsymbol{f}(0,z) = [A_1(z)]^{-1}[I - A_2(z)]\boldsymbol{F}$ (or $\boldsymbol{f}(0,z) = [I - A_3(z)]^{-1}A_4(z)\boldsymbol{F}$);
4: **return** the component of $\boldsymbol{f}(0,z)$ corresponding to the initial state of $\mathcal{A}$.

where $\begin{pmatrix} A_1(z) & A_2(z) \\ A_3(z) & A_4(z) \end{pmatrix} = \exp\left[z \begin{pmatrix} -M_S & -M_T \\ M_T & M_S \end{pmatrix}\right]$. 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 rows of (14) permits to express $\boldsymbol{f}(0,z)$ with respect to $\boldsymbol{F}$:

$$\begin{aligned} \boldsymbol{f}(0,z) &= [A_1(z)]^{-1}[I - A_2(z)]\boldsymbol{F} \\ \boldsymbol{f}(0,z) &= [I - A_3(z)]^{-1}A_4(z)\boldsymbol{F} \end{aligned} \tag{15}$$

Finally the coordinate of the column vector $\boldsymbol{f}(0,z)$ associated to the initial state gives the expected VGF. To sum up we have:

Theorem 3. *Given a regular language $L \subseteq \{U,D\}^*$, one can compute the exponential generating function $EGF(\mathfrak{p}^{-1}(L))(z)$ 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_T and M_S are the simplest possibles. Exponentiation of matrices is implemented in most of computer algebra systems.

4.2 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 2 and a sorting algorithm.

Theorem 4. *Let $L \subseteq \{U,D\}^*$ and $\mathbb{L}''$ be the timed semantic of a S-T-encoding of type l (for some $l \in \{U,D\}$) of a prolongation of L. The following algorithm permits to achieve a uniform sampling of permutation in $\mathfrak{p}^{-1}(L_{n-1})$. i.e. For $\sigma \in \mathfrak{p}_n^{-1}(L)$, the probability that the permutation σ is returned is $1/|\mathfrak{p}^{-1}(L_{n-1})|$.*

1. *Choose uniformly an n-length timed word $(\boldsymbol{t}, w) \in \mathbb{L}''_n$ using Algorithm 2;*
2. *Return $\mathtt{sort}(\phi_{\mathfrak{st}_l^{-1}(w)}(\boldsymbol{t}))$.*

Algorithm 2 Recursive uniform sampler of timed words

$x_0 \leftarrow 0$; $q_0 \leftarrow$ initial state;
for $k = 1$ to n **do**
  Compute $m_k = v_{q_{k-1},n-(k-1)}(x_{k-1})$ and $p_S = \int_{x_{k-1}}^{1} v_{\delta(q_{k-1},S),n-k}(y)dy/m_k$;
  $b \leftarrow \texttt{BERNOULLI}(p_S)$; (return 1 with probability p_S and 0 otherwise)
  if $b = 1$ **then**
    $w_k \leftarrow S$; $q_k \leftarrow \delta(q_{k-1}, S)$;
    $r \leftarrow \texttt{RAND}([0,1])$; (return a number uniformly sampled in $[0,1]$)
    $t_k \leftarrow$ the unique solution in $[0, 1-x_{k-1}]$ of $\frac{1}{m_k p_S} \int_{x_{k-1}}^{x_{k-1}+t_k} v_{q_k,n-k}(y)dy - r = 0$;
    $x_k \leftarrow x_{k-1} + t_k$;
  else
    $w_k \leftarrow T$; $q_k \leftarrow \delta(q_{k-1}, T)$;
    $r \leftarrow \texttt{RAND}([0,1])$; (return a number uniformly sampled in $[0,1]$)
    $t_k \leftarrow$ the unique solution in $[0, 1-x_{k-1}]$ of $\frac{1}{m_k(1-p_S)} \int_{0}^{t_k} v_{q_k,n-k}(y)dy - r = 0$;
    $x_k \leftarrow t_k$;
  end if
end for
return $(t_1, w_1)(t_2, w_2)\dots(t_n, w_n)$

Uniform sampling of timed words. Recursive formulae (16) and (17) 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] for a recall and an improvement of the recursive method in the context of random generation of word of a regular language; see also reference therein for application of the recursive method to more general combinatoric classes). 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 word. The languages $L_{q,n}(x)$ can be recursively defined as follow: $L_{q,0}(x) = \epsilon$ if $q \in F$ and $L_{q,0} = \emptyset$ otherwise;

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

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

$$v_{q,n+1}(x) = \int_x^1 v_{\delta(q,S),n}(y)dy + \int_0^{1-x} v_{\delta(q,T),n}(y)dy. \quad (17)$$

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: $\texttt{Vol}(\mathbb{L}''_n) = v_{q_0,n}(0)$ where q_0 is the initial state of the S-T automaton.

Theorem 5. *Algorithm 2 is a uniform sampler of timed words of $\mathbb{L}''_n$ i.e. for all volume measurable subset $A \subseteq \mathbb{L}''_n$, the probability that the returned timed word falls in A is $\texttt{Vol}(A)/\texttt{Vol}(\mathbb{L}''_n)$.*

Some comments about the algorithm. Algorithm 2 requires a precomputing of all functions $v_{q,k}$ for $q \in Q$ and $k \leq n$ done by Algorithm 3 below (see also proposition 4 for the complexity). The expressions in lines 8 and 13 are polynomials increasing in $[x, 1]$ (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.

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* $0(|Q|n^2)$. *Its bit space complexity is* $0(|Q|n^3)$.

Proof. The polynomial $v_{q,m}$ is of degree m, it has $O(m)$ coefficients. Therefore the time and space complexity are $0(\sum_{m=1}^{n} |Q|m) = O(|Q|n^2)$.

Magnitudes of coefficients of $v_{q,m}$ behave like $2^{m\mathcal{H}}$ where $\mathcal{H}$ is the entropy of 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.

5 Discussion and perspectives

We have stated and solved the problem of counting and uniform sampling of permutations with signature in a given regular language of signatures. The timed semantic of such a language is a particular case of timed regular 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 of signatures. For this we shall use as in our previous works [2,4] volume of languages parametrized both by a starting and an ending state.

Volumes and languages parametrized both by a starting and an ending state would also be useful to gain a linear factor in the time and space complexity of the preprocessing stage of the present setting (Algorithm 3). Indeed they are needed to adapt the divide and conquer algorithm of [5].

Our work can also benefit to timed automata community. 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]. A uniform sampler of timed word would

be useful to solve the proportional model checking problem introduced in our previous work [4].

A toy implementation of the algorithms is available on-line `http://www.liafa.univ-paris-diderot.fr/~nbasset/sage/sage.htm`.

Proofs and examples are given in the appendix

Acknowledgements.

I thank Eugene Asarin, Aldric Degorre and Dominique Perrin for sharing motivating discussions.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126, 183–235 (1994)
2. Asarin, E., Basset, N., Degorre, A., Perrin, D.: Generating functions of timed languages. In: MFCS. pp. 124–135. LNCS 7464 (2012)
3. Asarin, E., Degorre, A.: Volume and entropy of regular timed languages: Analytic approach. In: FORMATS. pp. 13–27. LNCS 5813 (2009)
4. Basset, N.: A maximal entropy stochastic process for a timed automaton (2013), to appear in ICALP'13
5. Bernardi, O., Giménez, O.: A linear algorithm for the random sampling from regular languages. Algorithmica 62(1-2), 130–145 (2012)
6. Bouyer, P., Petit, A.: A Kleene/Büchi-like theorem for clock languages. Journal of Automata, Languages and Combinatorics 7(2), 167–186 (2002)
7. Flajolet, P., Sedgewick, R.: Analytic combinatorics. Camb. Univ. press (2009)
8. Hibi, T., Li, N.: Unimodular equivalence of order and chain polytopes. arXiv preprint arXiv:1208.4029 (2012)
9. Stanley, R.P.: A survey of alternating permutations. Contemp. Math 531, 165–196 (2010)
10. Stanley, R.: Two poset polytopes. Discrete & Computational Geometry 1(1), 9–23 (1986)
11. Szpiro, G.G.: The number of permutations with a given signature, and the expectations of their elements. Discrete Mathematics 226(1), 423–430 (2001)

Appendix

A Some proof details

Proof of Proposition 1

Let $w = ul$ i.e. for all $i \in [n-1]$ $w_i = u_i$ and $w_n = l$. $L_{ul} \subseteq \mathcal{C}(u)$) Let $(t_1, \ldots, t_n) \in L_w$ i.e. there exist value of clocks x_k^a ($a \in \{U, D\}, k \in [n]$) such that $x_0^U = x_0^D = 0$ and $(x_{k-1}^U, x_{k-1}^D) \xrightarrow{(t_k, w_k)} (x_k^U, x_k^D) \in \mathcal{L}(w_k)$. Let $i < j \leq n$ and $a \in \{U, D\}$ 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.
$\mathcal{C}(u) \subseteq L_{ul}$) Let $(t_1, \ldots, t_n) \in \mathcal{C}(u)$. We show inductively that for every $a \in \{U, D\}$, the condition $x_{j-1}^a + t_j \leq 1$ is satisfied and thus that x_j^a can be defined ($x_j^a = x_{j-1}^a + t_j$ if $w_j = a$ and $x_j^a = 0$ 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.

Proof of proposition 2

The function ϕ_{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 $\phi_{ul}(\boldsymbol{t}) = \boldsymbol{\nu}$ iff $\boldsymbol{\nu}^\top = M_{ul}\boldsymbol{t}^\top + \boldsymbol{b}$ with for all $j \in [n]$: if $w_j = U$ (resp. $w_j = D$) then the j^{th} rows of the matrix M_{ul} as 1 (resp. -1) between coordinates i and j included and the j^{th} rows of $\boldsymbol{b}$ 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 we prove that $\boldsymbol{\nu} = \phi_{ul}\boldsymbol{t}$ belongs to $\mathcal{O}_n(u)$ for $\boldsymbol{t} \in \mathcal{C}(u)$. We show that the two conditions (C-1) and (C-2) below are equivalents. Then, we will be done since (C-1) is the definition of $(t_1, \cdots, t_n) \in \mathcal{C}(u)$ while (C-2) is equivalent to $\nu_1, \ldots, \nu_n \in \mathcal{O}(u)$:

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

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

Proof of Theorem 4

For all $\sigma \in \mathtt{p}_n^{-1}(L)$, the probability $p(\sigma)$ that the output is σ is the probability to choose a timed word $(\boldsymbol{t}, w)$ such that $\mathtt{sort}[\phi_{\mathtt{st}^{-1}(w)}(\boldsymbol{t})] = \sigma$. Since the timed words are uniformly sampled this probability is equal to $\mathtt{Vol}(\{(\boldsymbol{t}, w) \mid \mathtt{sort}[\phi_w(\boldsymbol{t})] = \sigma\})/\mathtt{Vol}(\mathbb{L}''_n)$ which is equal to $\mathtt{Vol}(\{\boldsymbol{\nu} \mid \mathtt{sort}(\boldsymbol{\nu}) = \sigma\})/\mathtt{Vol}(\mathbb{L}''_n)$ since the mapping $(\boldsymbol{t}, w) \mapsto \phi_{\mathtt{st}^{-1}(w)}(\boldsymbol{t})$ is a volume preserving transformation. The numerator is the volume of the order simplex associated to σ which is $\mathtt{Vol}(\mathcal{O}(\sigma)) = 1/n!$; the denominator $\mathtt{Vol}(\mathbb{L}''_n)$ is $|\mathtt{p}^{-1}(L_{n-1})|/n!$ by virtue of Theorem 2. We get the expected result $p(\sigma) = (1/n!)/(|\mathtt{p}^{-1}(L_{n-1})|/n!) = 1/|\mathtt{p}^{-1}(L_{n-1})|$. □

More details about Algorithm 2, sketch of proof of Theorem 5.

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 all timed words of $\mathbb{L}''$ i.e. $p[(t_1, w_1) \cdots (t_n, w_n)] = 1/\mathtt{Vol}(\mathbb{L}'')$.

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

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

We show that

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

The choice of (t_k, w_k) is done in two steps: we chose first w_k (and thus q_k) and then t_k. We write this

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

We have that $p_k[S \mid q_{k-1}, x_{k-1}] = p_S$ and $p_k[T \mid q_{k-1}, x_{k-1}] = 1 - p_S$ since $b = 1$ iff $w_k = S$.

In both cases ($b = 0$ or 1) the delay t_k is sampled using the so called inverse sampling method. This method state that to sample a random variable according to a probability density function (PDF) $p(t)$ (here $p(t) = p_k[t \mid q_k, x_{k-1}]$) it suffices to uniformly sampled 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 [8] (CDF) associated to p.

[8] Its inverse (t function of r) is known as the quantile function.

When $b = 1$ (and thus $w_k = S$), the CDF used in the algorithm is $t \mapsto \int_0^t v_{q_k,n-k}(x_{k-1} + t')/p_S m_k dt'$. Its corresponding PDF is $p_k[t_k \mid q_k, x_{k-1}] = v_{q_k,n-k}(x_{k-1}+t_k)/p_S m_k = m_{k+1}/p_S m_k$. Multiplying by $p_k[S \mid q_{k-1}, x_{k-1}] = p_S$ we get (19).

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

Plugging (19) in (18), we get the expected result:

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

B Examples

In section B.1 we show how Algorithm 1 applies to the classical example of alternating permutations. In section B.2 we apply this algorithm to what we call up-up-down-down permutations. In section B.3 we treat the running example given in the article.

B.1 The alternating permutations

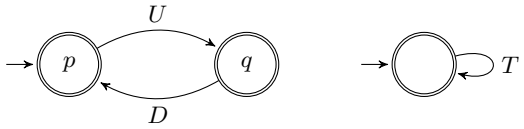

Fig. 1. An automaton for $(UD)^*(U + \epsilon)$ and its S-T encoding of type D

The class of alternating permutation is[9] $\mathtt{Alt} = \mathfrak{S}_0 \cup \mathtt{p}^{-1}[(UD)^*(U + \epsilon)]$. It is well known since the 19^{th} century and the work of Désiré André that

$$EGF(\mathtt{Alt})(z) = \tan(z) + \sec(z) \quad (\sec(z) = 1/\cos(z)).$$

Several different proof of this results can be found in [9]. Here we give a novel proof based on the application of Algorithm 1 on $(UD)^*(U + \epsilon)$.

A prolongation of $(UD)^*(U+\epsilon)$ is $(UD)^*(UD+U)$. We add ϵ to the language to add 1 to its VGF, indeed

$$EGF(\mathtt{Alt})(z) = 1 + VGF[(UD)^*(UD + U)](z) = VGF[(UD)^*(U + \epsilon)](z)$$

The S-T encoding of type D of $(UD)^*(\epsilon + U)$ is just S^* which is recognized by the one loop automaton depicted on the right of figure 1 . Thus $M_S = (1)$,

[9] Abusing the notation, we confuse regular expressions with the regular language they express

$M_T = (0)$ 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: $M^0 = I_2$ $M^2 = -I_2$, $M^2 = -M$, $M^3 = I_2$,$M^4 = M$,... Then for all $k \geq 0$:

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

Therefore $\exp(zM) = \sum_{n\in\mathbb{N}} z^n M^n/n! = \begin{pmatrix} \cos(z) & -\sin(z) \\ \sin(z) & \cos(z) \end{pmatrix}$ By definition $A_1(z) = \cos(z)$, $A_2(z) = -\sin(z)$ and we thus we can conclude:

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

B.2 The up-up-down-down permutations

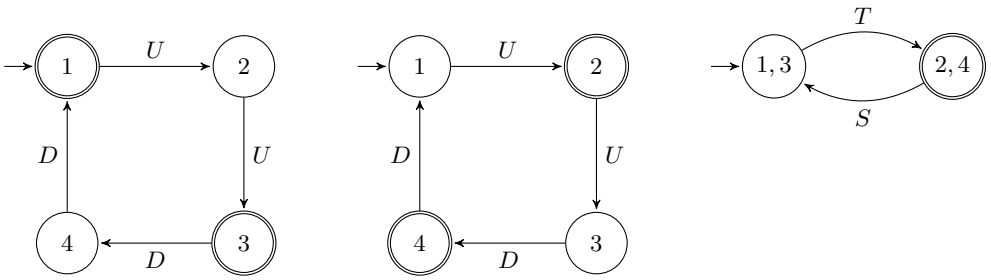

Fig. 2. From left to right: automata for L, L', $\texttt{st}_D(L')$

Alternating permutations (treated above) of odd lengths are sometimes called up-down permutations. Here we compute the exponential generating function of up-up-down-down permutations i.e. those with signature in $L = (UUDD)^*(UU+\epsilon)$. The language $L' = (UUDD)^*(UUD + U)$ is an extension of L. Its S-T-encoding of type D is $\texttt{st}_D(L') = (TS)^*T$. This latter languages is recognized by the S-T-automaton depicted on figure 2. Its adjacency matrices are $M_S = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$, $M_T = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ and the row vector of final state is $\boldsymbol{F} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Let $M = \begin{pmatrix} -M_S & -M_T \\ M_T & M_S \end{pmatrix}$. One more time computation of $\exp(zM)$ is easy since M is unipotent[10]:

$$M = \begin{pmatrix} 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}; M^2 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}; M^3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix}; M^4 = I_4.$$

[10] This is the case for all cyclic automaton

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 function:

$$\begin{array}{ll} f_0(z) = [\cosh(z) + \cos(z)]/2; & f_1(z) = [\sinh(z) + \sin(z)]/2; \\ f_2(z) = [\cosh(z) - \cos(z)]/2; & f_3(z) = [\sinh(z) - \sin(z)]/2. \end{array}$$

We have

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

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:

$$f(z) = f_p(z) = (\sinh z - \sin z + \sin(z)\cosh z + \sinh(z)\cos z)/(1 + \cos(z)\cosh z).$$

B.3 The running example detailed

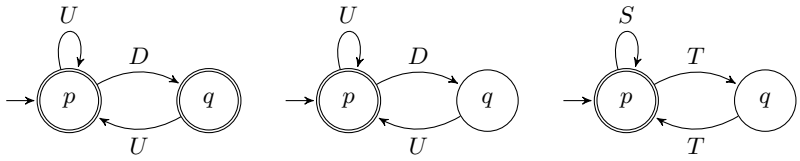

Fig. 3. From left to right automata for L^{ex}, $L^{ex\prime} = L^{ex}.\{U\} \cup \{\epsilon\}$, $L^{ex\prime\prime} = \mathtt{st}_U(L^{ex\prime})$

Consider again the class C^{ex} of permutation without two consecutive downs. Then $C^{ex} = \mathfrak{S}_0 \cup \mathtt{p}^{-1}((U+DU)^*(D+\epsilon))$. A prolongation of $(U+DU)^*(D+\epsilon)$. is $(U+DU)^*U$. As for alternating permutations we add the word ϵ to this language to add 1 to the final generating function, thus we get the language $(U + DU)^*$ recognized by the automaton depicted in the middle of figure 3. Its S-T encoding of type U is $(S + TT)^*$ which is recognized by the automaton depicted on the right of figure 3. Its adjacency matrices are $M_S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $M_T = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and the row vector of final state is $\boldsymbol{F} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Let $M = \begin{pmatrix} -M_S & -M_T \\ M_T & M_S \end{pmatrix}$. We will

solve directly the differential equation (10) with boundary condition (11), i.e. the system

$$\frac{\partial f_p}{\partial x}(x,z) = -zf_p(x,z)dy - zf_q(1-x,z)dy; \tag{21}$$
$$\frac{\partial f_q}{\partial x}(x,z) = -zf_p(1-x,z). \tag{22}$$

with boundary conditions $f_p(1,z) = 1; f_q(1,z) = 0$ Equation (21) taken in $x = 1$ 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 condition

$$f_p(1,z) = 1; \tag{23}$$
$$\frac{\partial f_p}{\partial x}(1,z) = -zf_p(0,z). \tag{24}$$

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

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

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

$$a(z)\cos(\sqrt{3}z/2) + b(z)\sin(\sqrt{3}z/2) = e^{z/2} \tag{26}$$
$$a(z) + \sqrt{3}b(z) = 0; \tag{27}$$

Solving this system we get:

$$a(z) = e^{z/2}\sqrt{3}/d(z) \tag{28}$$
$$b(z) = \left[e^{z/2}/2 - \cos(z\sqrt{3}/2)\right]/d(z) \tag{29}$$

where $d(z) = \cos(z\sqrt{3}/2)\sqrt{3} - \sin(z\sqrt{3}/2) = [3\cos^2(z\sqrt{3}/2) - \sin^2(z\sqrt{3}/2)]/[\cos(z\sqrt{3}/2)\sqrt{3} + \sin(z\sqrt{3}/2)]$. The numerator of this fraction is equal to $4\cos^2(z\sqrt{3}/2) - 1 = (2\cos^2(z\sqrt{3}/2) - 1)(2\cos^2(z\sqrt{3}/2) + 1)$ Finally the generating function is

$$EGF(C^{ex})(z) = f_p(0,z) = a(z) = \frac{3\cos(z\sqrt{3}/2) + \sqrt{3}\sin(z\sqrt{3}/2)}{[2\cos(z\sqrt{3}/2) - 1][2\cos(z\sqrt{3}/2) + 1]}e^{z/2}$$