Compter et générer aléatoirement des permutations décrites par un langage régulier[1]

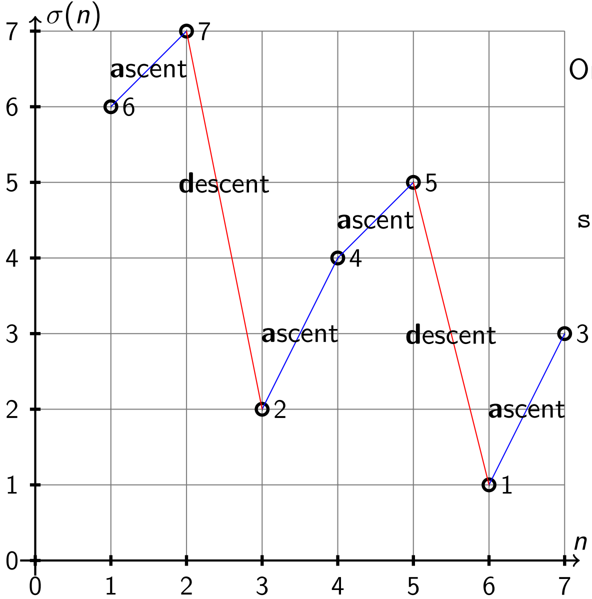
Nicolas Basset

University of Oxford

Journée ALÉA 2014

[1] sera présenté à LATIN 2014

Signature of a permutation



One line notation:
$\sigma = 6724512$

Signature:
$\mathrm{sg}(\sigma) = \textbf{adaada}$

Two problem statements

Given a regular language $L \subseteq \{\mathbf{a}, \mathbf{d}\}^*$, we are interested in

$$\mathrm{sg}^{-1}(L) = \{\sigma \mid \mathrm{sg}(\sigma) \in L\}.$$

Problem 1: Enumeration

Design an algorithm that compute a closed form formula for the exponential generating function:

$$F_L(z) = \sum_{\sigma,\ \mathrm{sg}(\sigma) \in L} \frac{z^{|\sigma|}}{|\sigma|!} = \sum_{n \geq 1} \alpha_n(L) \frac{z^n}{n!}$$

where $\alpha_n(L) = |\{\sigma \in \mathfrak{S}_n \mid \mathrm{sg}(\sigma) \in L\}|$

Problem 2: Uniform sampling

Construct a uniform random sampler for $\{\sigma \in \mathfrak{S}_n \mid \mathrm{sg}(\sigma) \in L\}$. That is $Prob(output = \sigma) = \frac{1}{\alpha_n(L)}$.

Examples

Examples of closed form formula for F_L

- Alternating permutations: $F_{(\mathbf{ad})^*(\mathbf{a}+\epsilon)} = \tan(z) + \sec(z) - 1$
- No two consecutive descents:
  $F_{(\mathbf{a}+\mathbf{da})^*(\mathbf{d}+\epsilon)}(z) = \frac{3\cos(z\sqrt{3}/2)+\sqrt{3}\sin(z\sqrt{3}/2)}{[2\cos(z\sqrt{3}/2)-1][2\cos(z\sqrt{3}/2)+1]} e^{z/2} - 1$
- Up-up-down-down permutations :
  $F_{(\mathbf{aadd})^*(\mathbf{aa}+\epsilon)} = \frac{\sinh z - \sin z + \sin(z)\cosh z + \sinh(z)\cos z}{1+\cos(z)\cosh z}$
- Even number of descents (homework).

A permutation without 2 consecutive descents ($n = 100$)

$$[75, 76, 7, 72, 81, 64, 77, 55, 97, 15, 95, 18, 98, 32, 93, 17, 67, 12, 49, 85, 22, 50, 21, 68, 57, 87, 27, 41, 52, 61, 91, 26, 30, 59, 33, 73, 5, 54, 39, 43, 28, 44, 14, 62, 11, 80, 40, 47, 45, 66, 56, 69, 86, 19, 78, 90, 37, 71, 51, 99, 13, 48, 4, 34, 83, 100, 1, 6, 46, 82, 9, 35, 60, 29, 84, 20, 58, 79, 2, 38, 96, 10, 23, 88, 3, 53, 94, 36, 89, 16, 31, 24, 63, 8, 74, 42, 65, 70, 92, 25]$$

Related work

Descent pattern avoidance [Ehrenborg, Jung 2013]

Finite set F of forbidden words $\rightarrow$ language of finite type
$X_F = \{w \in \{\mathbf{a}, \mathbf{d}\}^* \mid w_i \cdots w_j \notin F\}$
Descent pattern avoidance: $\mathtt{sg}^{-1}(X_F) = \{\sigma \mid \sigma \text{ avoids } F\}$.

Example

$X_{\mathbf{aa},\mathbf{dd}}$: alternating permutations $\sigma_1 < \sigma_2 > \sigma_3 < \dots$.

Related work

Descent pattern avoidance [Ehrenborg, Jung 2013]

Finite set F of forbidden words $\to$ language of finite type
$X_F = \{w \in \{\mathbf{a}, \mathbf{d}\}^* \mid w_i \cdots w_j \notin F\}$
Descent pattern avoidance: $\mathrm{sg}^{-1}(X_F) = \{\sigma \mid \sigma \text{ avoids } F\}$.

Example

$X_{\mathbf{aa},\mathbf{dd}}$: alternating permutations $\sigma_1 < \sigma_2 > \sigma_3 < \ldots$.

Recall: language of finite type cannot express all regular languages: e.g. even number of descents.

Related work

Descent pattern avoidance [Ehrenborg, Jung 2013]

Finite set F of forbidden words $\to$ language of finite type
$X_F = \{w \in \{\mathbf{a}, \mathbf{d}\}^* \mid w_i \cdots w_j \notin F\}$
Descent pattern avoidance: $\mathrm{sg}^{-1}(X_F) = \{\sigma \mid \sigma \text{ avoids } F\}$.

Example

$X_{\mathbf{aa},\mathbf{dd}}$: alternating permutations $\sigma_1 < \sigma_2 > \sigma_3 < \dots$.

Recall: language of finite type cannot express all regular languages: e.g. even number of descents.

Prescribed descent set ([Marchal 2013])

Random sampling when $L = w$ with $w \in \{\mathbf{a}, \mathbf{d}\}^*$.
Generating function when $L = \mathrm{Pref}(w^*)$ with $w \in \{\mathbf{a}, \mathbf{d}\}$ (i.e. for cyclic automata).

Methodology

- Geometric interpretation of the two problems.
- Reduction to volumetry of some timed language.
- Solutions based on **volume equations** for timed language.

A geometric interpretation

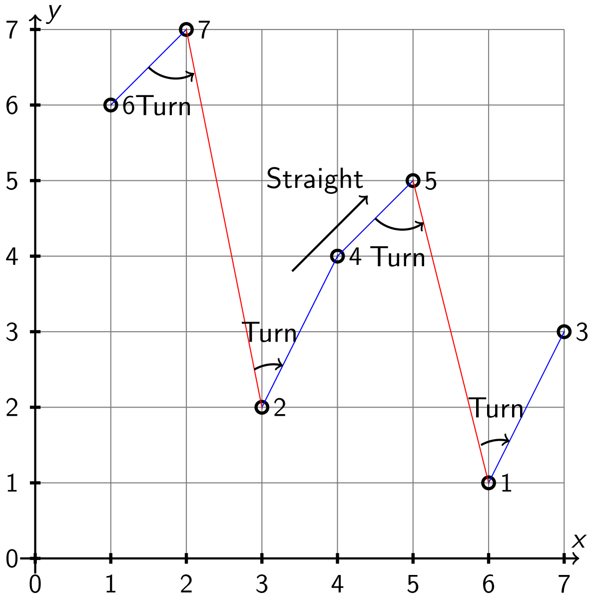
Order polytopes of permutation and words

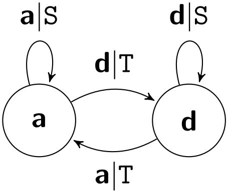
- $\mathcal{O}(\sigma) = \{\vec{\nu} \in [0,1]^n \mid \nu_i < \nu_j \text{ iff } \sigma_i < \sigma_j \text{ for } i \neq j\}$.
- Remark $[0,1]^n = \cup_{\sigma \in \mathfrak{S}_n} \mathcal{O}(\sigma)$ and $\mathtt{Vol}\,\mathcal{O}(\sigma) = 1/n!$
- $\mathcal{O}(u) =_{def} \sqcup_{\mathrm{sg}(\sigma)=u} \mathcal{O}(\sigma)$ e.g.
  $\mathcal{O}(\mathbf{daa}) = \mathcal{O}(2134) \sqcup \mathcal{O}(3124) \sqcup \mathcal{O}(4123)$.
- $\mathtt{Vol}(\mathcal{O}(u)) = |\{\sigma \mid \mathrm{sg}(\sigma) = u\}|/n!$.

Volume Generating Function

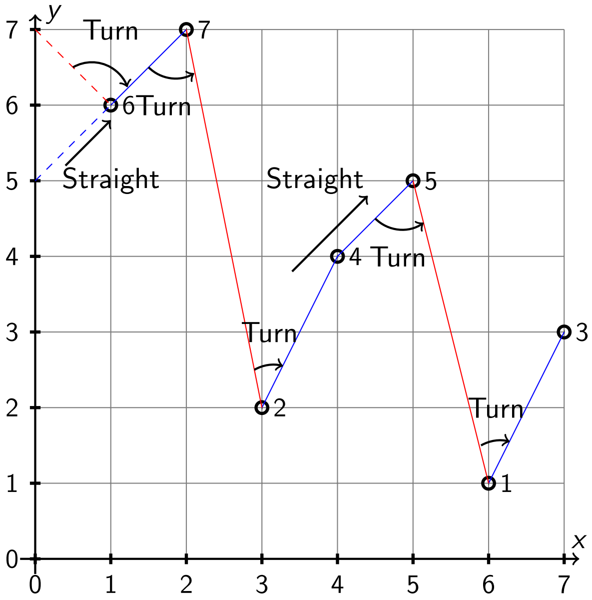
$$g_L(z) = \sum_{\sigma \mid \mathrm{sg}(\sigma) \in L} \frac{z^{|\sigma|}}{|\sigma|!} = \sum_{u \in L} \mathtt{Vol}(\mathcal{O}(u)) z^{|u|+1}.$$

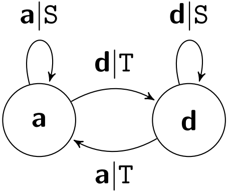
The straight-turn encoding



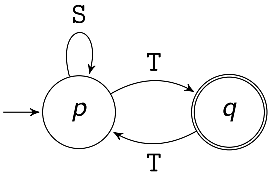


The straight-turn encoding





Adding time and clocks to words.



- a word: $\mathtt{SST} \in \{\mathtt{S}, \mathtt{T}\}^*$
- a timed word $(0.5, \mathtt{S})(0.3, \mathtt{S})(0.1, \mathtt{T}) \in ([0, 1] \times \{\mathtt{S}, \mathtt{T}\})^*$
- a clock word $0 \xrightarrow{(0.5,\mathtt{S})} 0.5 \xrightarrow{(0.3,\mathtt{S})} 0.8 \xrightarrow{(0.1,\mathtt{T})} 0.1$

The straight and turn timed transitions

Straight: $x \xrightarrow{(t,S)} x + t$ if $x + t \leq 1$

Turn: $x \xrightarrow{(t,T)} t$ if $x + t \leq 1$

The timed semantic of a language $\subseteq \{\mathtt{S}, \mathtt{T}\}^*$

$$0 \xrightarrow{(0.5,\mathtt{S})} 0.5 \xrightarrow{(0.3,\mathtt{S})} 0.8 \xrightarrow{(0.1,\mathtt{T})} 0.1 = 0 \xrightarrow{(0.5,0.3,0.1)\mathtt{SST}} 0.1$$

Timed semantics of $L'' \subseteq \{\mathtt{S}, \mathtt{T}\}^*$

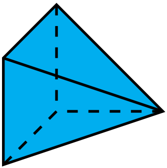
- The timed polytope associated to $w \in \{\mathtt{S}, \mathtt{T}\}^*$ is
  $P_w = \{\vec{t} \mid 0 \xrightarrow{(\vec{t},w)} y \text{ for some } y \in [0,1]\}$.
  e.g. $(0.5, 0.3, 0.1) \in P_{\mathtt{SST}}$.
- The timed semantics $L'' \subseteq \{\mathtt{S}, \mathtt{T}\}^*$ is

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

Volume generating function of $L'' \subseteq \{\mathtt{S}, \mathtt{T}\}^*$

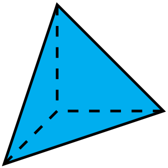
Two timed polytopes

$P_{\mathtt{TTT}}$



$t_1 + t_2 \leq 1$ and $t_2 + t_3 \leq 1$

$P_{\mathtt{SSS}}$:



$t_1 + t_2 + t_3 \leq 1$

- Recall $\mathbb{L}'' = \cup_{w \in L} P_w \times \{w\}$.
- $\mathtt{Vol}(\mathbb{L}''_n) =_{def} \sum_{w \in L''_n} \mathtt{Vol}(P_w)$
- Volume Generating Function of $\mathbb{L}''$:
  $VGF(L'')(z) =_{def} \sum_{n \geq 0} \mathtt{Vol}(\mathbb{L}''_n) z^n = \sum_{w \in L''} \mathtt{Vol}(P_w) z^{|w|}$.

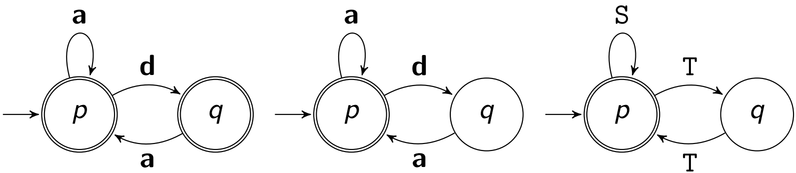
The key lemma

Two step bijection: prolongating and then encoding in $\{\mathtt{S},\mathtt{T}\}^*$

$$h : \begin{cases} L & \to & L'' =_{def} h(L) \\ u & \mapsto & w \quad \text{encoding of } u\mathbf{a} \text{ in } \{\mathtt{S},\mathtt{T}\}^*. \end{cases}$$

Remark: easy to compute when a DFA for L'' from a DFA for L.

No two consecutive descents: L, $La \cup \{\epsilon\}$, $L'' \cup \{\epsilon\}$



Key lemma

For every $w = h(u) \in L''$, there is a volume preserving transformation $\phi_w : \mathbb{L}''_w \to \mathcal{O}(u)$ (computable in $O(|w|)$).

Reducing the two problems

Reduction for Problem 1 (exponential generating function)

$$F_L(z) = \sum_{u \in L} \mathtt{Vol}(\mathcal{O}(u)) z^{|u|+1} = \sum_{w \in L''} \mathtt{Vol}(\mathbb{L}''_w) z^{|w|} = VGF(L'')(z).$$

(Recall: volume preserving transformation $\phi_w : \mathbb{L}''_w \to \mathcal{O}(u)$.)

Reduction for Problem 2 (uniform sampling)

1. Choose uniformly an n-length timed word $(\vec{t}, w) \in \mathbb{L}''_n$;
2. compute $\vec{\nu} = \phi_w(\vec{t}) \in \mathcal{O}_n(L)$;
3. return σ such that $\vec{\nu} \in \mathcal{O}(\sigma)$ (using a sort).

Reducing the two problems

Reduction for Problem 1 (exponential generating function)

$$F_L(z) = \sum_{u \in L} \mathtt{Vol}(\mathcal{O}(u)) z^{|u|+1} = \sum_{w \in L''} \mathtt{Vol}(\mathbb{L}''_w) z^{|w|} = VGF(L'')(z).$$

(Recall: volume preserving transformation $\phi_w : \mathbb{L}''_w \to \mathcal{O}(u)$.)

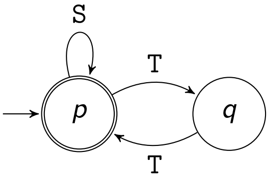
Reduction for Problem 2 (uniform sampling)

1. Choose uniformly an n-length timed word $(\vec{t}, w) \in \mathbb{L}''_n$;
2. compute $\vec{\nu} = \phi_w(\vec{t}) \in \mathcal{O}_n(L)$;
3. return σ such that $\vec{\nu} \in \mathcal{O}(\sigma)$ (using a sort).

Now it suffices to solve the problems for timed automata.

Language and VGF equations

No two consecutive descents



parametrized language equations

$$
\begin{aligned}
L_p(x) &= \cup_{t \leq 1-x}(t, \mathtt{S})L_p(x+t) \cup \quad \cup_{t \leq 1-x}(t, \mathtt{T})L_q(t) \cup \epsilon \\
L_q(x) &= \quad \cup_{t \leq 1-x}(t, \mathtt{T})L_p(t)
\end{aligned}
$$

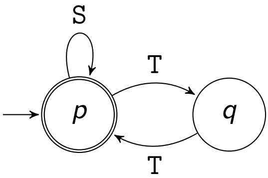
parametrized VGF

$$
\begin{aligned}
f_p(x,z) &= z \int_{t \leq 1-x} f_p(x+t, z)dt + \quad z \int_{t \leq 1-x} f_q(t,z)dt + 1 \\
f_q(x,z) &= \quad z \int_{t \leq 1-x} f_p(t,z)dt
\end{aligned}
$$

The matrix notation

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

No two consecutive descents



$$M_{\mathtt{S}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},\ M_{\mathtt{T}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \vec{F} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Solving the equation

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

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

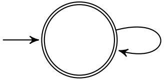
$$\begin{pmatrix} \vec{f}(1,z) \\ \vec{f}(0,z) \end{pmatrix} = \exp\left[z \begin{pmatrix} -M_{\mathrm{S}} & -M_{\mathrm{T}} \\ M_{\mathrm{T}} & M_{\mathrm{S}} \end{pmatrix} \right] \begin{pmatrix} \vec{f}(0,z) \\ \vec{f}(1,z) \end{pmatrix} \text{ and } \vec{f}(1,z) = \vec{F}$$

An algorithm to compute $F_L(z) = f_{q_0}(0,z)$

1. Compute $\begin{pmatrix} A_1(z) & A_2(z) \\ A_3(z) & A_4(z) \end{pmatrix} =_{def} \exp\left[z \begin{pmatrix} -M_{\mathrm{S}} & -M_{\mathrm{T}} \\ M_{\mathrm{T}} & M_{\mathrm{S}} \end{pmatrix} \right]$;
2. return $F_L(z)$ the component of $\vec{f}(0,z) = [A_1(z)]^{-1}[I - A_2(z)]\vec{F} = [I - A_3(z)]^{-1}A_4(z)\vec{F}$ corresponding to the initial state q_0.

A classical example: the alternating permutations

e.g. $\sigma = 94738251$



Here $\vec{F} = M_{\mathtt{T}} = (1)$, $M_{\mathtt{S}} = (0)$.

$$\exp\left[z\begin{pmatrix} -M_{\mathtt{S}} & -M_{\mathtt{T}} \\ M_{\mathtt{T}} & M_{\mathtt{S}} \end{pmatrix}\right] = \exp\begin{pmatrix} 0 & -z \\ z & 0 \end{pmatrix} = \begin{pmatrix} \cos z & -\sin z \\ \sin z & \cos z \end{pmatrix}$$

(Recall $F_L(z) = [A_1(z)]^{-1}[I - A_2(z)]\vec{F} = [I - A_3(z)]^{-1}A_4(z)\vec{F}$)

$$F_L(z) = \frac{1+\sin z}{\cos z} = \frac{\cos z}{1-\sin z}$$

Generating timed word using the recursive method

Recursive language equations and volume equations

$$
\begin{aligned}
L_{p,n}(x) &= \cup_{t\leq 1-x}(t,\mathtt{S})L_{p,n-1}(x+t)\cup \quad \cup_{t\leq 1-x}(t,\mathtt{T})L_{q,n-1}(t) \\
v_{p,n}(x) &= \int_0^{1-x} v_{p,n-1}(x+t)dt + \quad \int_0^{1-x} v_{q,n-1}(t)dt
\end{aligned}
$$

How can one generate a timed word in $L_{p,n}(x)$?

Generating timed word using the recursive method

Recursive language equations and volume equations

$$
\begin{aligned}
L_{p,n}(x) &= \cup_{t\leq 1-x}(t,\mathtt{S})L_{p,n-1}(x+t)\cup \quad \cup_{t\leq 1-x}(t,\mathtt{T})L_{q,n-1}(t) \\
v_{p,n}(x) &= \int_0^{1-x} v_{p,n-1}(x+t)dt + \quad \int_0^{1-x} v_{q,n-1}(t)dt
\end{aligned}
$$

How can one generate a timed word in $L_{p,n}(x)$?

- Choose between $\mathtt{S}$ and $\mathtt{T}$ according to $(P_{\mathtt{S}}, 1-P_{\mathtt{S}})$ with $P_{\mathtt{S}} = \int_0^{1-x} v_{p,n-1}(x+t)dt / v_{p,n}(x)$.
- If $\mathtt{T}$ is chosen then choose t according to the density: $\frac{v_{q,n-1}(t)1_{t<1-x}}{\int_0^{1-x} v_{q,n-1}(t)dt}$.
- If $\mathtt{S}$ is chosen...

Generating timed word using the recursive method

Recursive language equations and volume equations

$$L_{p,n}(x) = \cup_{t\leq 1-x}(t,\mathtt{S})L_{p,n-1}(x+t) \cup \quad \cup_{t\leq 1-x}(t,\mathtt{T})L_{q,n-1}(t)$$
$$v_{p,n}(x) = \int_0^{1-x} v_{p,n-1}(x+t)dt + \quad \int_0^{1-x} v_{q,n-1}(t)dt$$

How can one generate a timed word in $L_{p,n}(x)$?

- Choose between S and T according to $(P_{\mathtt{S}}, 1-P_{\mathtt{S}})$ with $P_{\mathtt{S}} = \int_0^{1-x} v_{p,n-1}(x+t)dt / v_{p,n}(x)$.
- If T is chosen then choose t according to the density: $\frac{v_{q,n-1}(t)1_{t<1-x}}{\int_0^{1-x} v_{q,n-1}(t)dt}$.
- If S is chosen...
- Repeat recursively.

Generating timed word using the recursive method

Recursive language equations and volume equations

$$
\begin{aligned}
L_{p,n}(x) &= \cup_{t\le 1-x}(t,\mathtt{S})L_{p,n-1}(x+t) \cup \quad \cup_{t\le 1-x}(t,\mathtt{T})L_{q,n-1}(t) \\
v_{p,n}(x) &= \int_0^{1-x} v_{p,n-1}(x+t)dt + \quad \int_0^{1-x} v_{q,n-1}(t)dt
\end{aligned}
$$

How can one generate a timed word in $L_{p,n}(x)$?

- Choose between $\mathtt{S}$ and $\mathtt{T}$ according to $(P_{\mathtt{S}}, 1-P_{\mathtt{S}})$ with $P_{\mathtt{S}} = \int_0^{1-x} v_{p,n-1}(x+t)dt / v_{p,n}(x)$.
- If $\mathtt{T}$ is chosen then choose t according to the density: $\frac{v_{q,n-1}(t)1_{t<1-x}}{\int_0^{1-x} v_{q,n-1}(t)dt}$.
- If $\mathtt{S}$ is chosen...
- Repeat recursively.

Need precomputation of $v_{q,k}$, $q \in Q$, $k = 0..n$. Complexity polynomial: $O(|Q|n^2)$. The generation itself is linear.

Conclusion

What we have seen

1. Bijection: permutations $\leftrightarrow$ order simplices.
2. Volume preserving transformation between order polytopes and timed polytopes (=chain polytopes).
3. Solution of the problems using new kind of timed languages involving $\mathtt{S}$ and $\mathtt{T}$.

Further works

1. Improvement of the algorithms.
2. Precise growth rate of $\alpha_n(L)$.
3. Random generation based on maximal entropy stochastic process over runs of a timed automaton [ICALP'13].
4. Extension to non regular languages like context free languages ($S \to \varepsilon \mid \mathbf{a}S\mathbf{d}S$).

Bonus: periodic descent set (see also [Marchal], [Luck])

Periodic language $L = \mathrm{Pref}(w^*)$ with $w \in \{\mathbf{a}, \mathbf{d}\}$ iff recognized by a cyclic automaton with $p =_{def} |w|$ states.

$$M^{2p} = \begin{pmatrix} -M_{\mathrm{S}} & -M_{\mathrm{T}} \\ M_{\mathrm{T}} & M_{\mathrm{S}} \end{pmatrix}^{2p} = (-1)^p I_{2p} \quad \left(e.g. \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^2 = -I_2 \right)$$

Bonus: periodic descent set (see also [Marchal], [Luck])

Periodic language $L = \mathrm{Pref}(w^*)$ with $w \in \{\mathbf{a}, \mathbf{d}\}$ iff recognized by a cyclic automaton with $p =_{def} |w|$ states.

$$M^{2p} = \begin{pmatrix} -M_{\mathrm{S}} & -M_{\mathrm{T}} \\ M_{\mathrm{T}} & M_{\mathrm{S}} \end{pmatrix}^{2p} = (-1)^p I_{2p} \quad \left(e.g. \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^2 = -I_2 \right)$$

Theorem

$F_L(z) = R(g_{0,p}(z), \ldots, g_{2p-1,p}(z))$ with R a rational function and

$$g_{k,p}(z) = \sum_{m \geq 0} (-1)^{pm} \frac{z^{k+2pm}}{(k+2pm)!}$$

$(\exp(zM) = \sum_{k=0}^{2p-1} g_{k,p}(z) M^k$; $\vec{f}(0,z) = [A_1(z)]^{-1}[I - A_2(z)]\vec{F})$

$g_{0,1}(z) = \cos z$; $g_{1,1}(z) = \sin z$;
$g_{0,2}(z) = [\cosh z + \cos z]/2$; $g_{1,2}(z) = [\sinh z + \sin z]/2$;
$g_{2,2}(z) = [\cosh z - \cos z]/2$; $g_{3,2}(z) = [\sinh z - \sin z]/2$;