Sofic-Dyck shifts

Marie-Pierre Béal, Michel Blockelet
and Cătălin Dima

Université Paris-Est
Laboratoire d'informatique Gaspard-Monge UMR 8049
Laboratoire d'algorithmique, complexité et logique

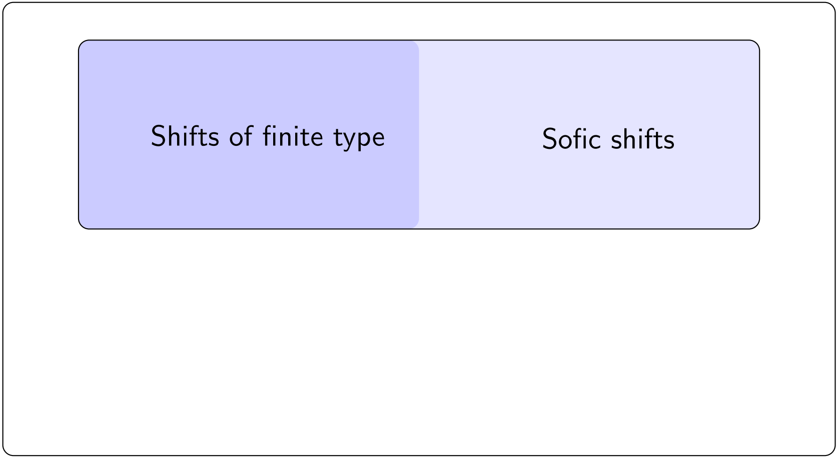
SDA2 GDR IM, LORIA 2014

Overview

- Background: shifts of finite type, sofic shifts
- Sofic-Dyck and finite-type-Dyck shifts
- Zeta functions
- The zeta function of a sofic-Dyck shift is $\mathbb{N}$-algebraic
- Decomposition theorem for finite-type-Dyck shifts

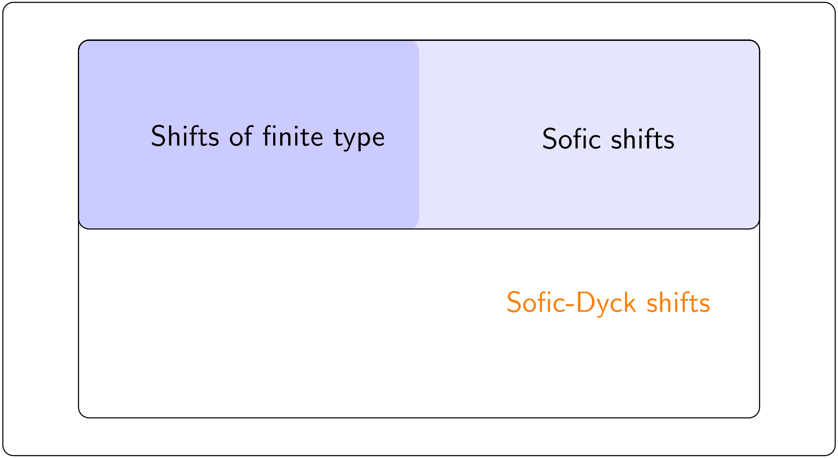
Shifts of sequences

Sets of bi-infinite sequences of symbols avoiding a given set of (finite) forbidden factors.



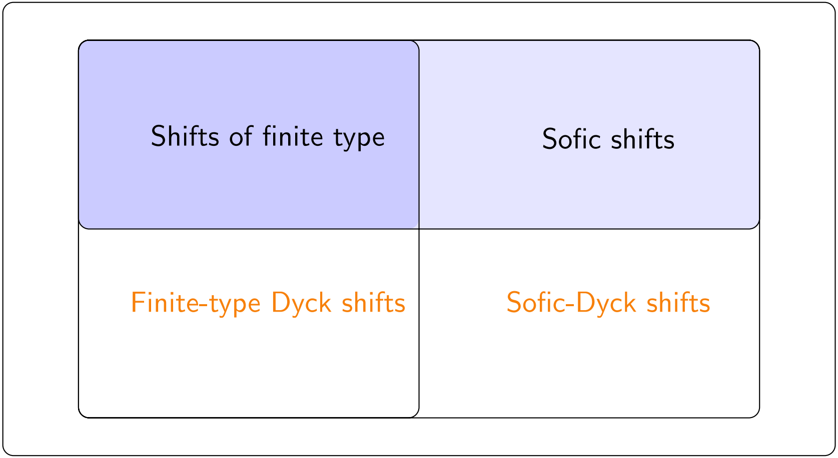
Shifts of sequences

Sets of bi-infinite sequences of symbols avoiding a given set of (finite) forbidden factors.



Shifts of sequences

Sets of bi-infinite sequences of symbols avoiding a given set of (finite) forbidden factors.



Background

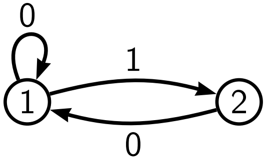
Dyck shifts, *Krieger et al.*
Markov-Dyck shifts, *Krieger and Matsumoto*
Extensions of Markov-Dyck shifts, *Inoue and Krieger*
Shifts presented by $\mathcal{R}$-graphs, *Krieger*

Contrained sequences of finite-type

A forbidden sequence:

$$\cdots 0100101010100\textcolor{red}{11}0001010 \cdots$$

Characterized by a finite set of forbidden blocks $\textcolor{red}{\{11\}}$.

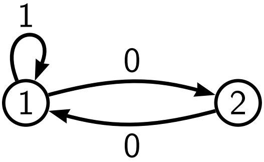


Sofic constraints

A forbidden sequence:

$$\cdots 01001010101001\textcolor{red}{10001}010 \cdots$$

Characterized by a regular set of forbidden blocks: an odd number of 0 between two 1 is forbidden.



Applications

- Coding for storage devices
- Magnetic and optical recording : *i.e.* $11, 101, 00000000$ are forbidden
- Flash memories : 101 or $(q-1, 0, q-1)$ is forbidden for q-ary cells.
- Flash memories : balanced sequences $+$ 101 forbidden (During reading, $n/2$ cells with lower voltage are read as 0 and $n/2$ cells with higher voltage are read as 1).

Beyond sofic constraints: Dyck constraints

Allowed sequences:

$\cdots [\, (\, (\,) \,) \,] \, [\,] \, [\, (\cdots$

$\cdots [\, (\, (\, [\, (\, [\, (\, [\, (\cdots$

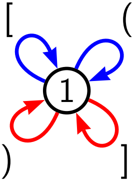
$\cdots (\,) \, (\,) \, (\,) \, (\,) \, (\,) \cdots$

Forbidden sequences:

$\cdots [\, (\, {\color{red}(} \, {\color{red}]} \,) \,] \, [\,] \, [\, (\cdots$

$\cdots [\, (\, (\, {\color{red}[} \, (\, {\color{red}]} \, (\, [\, (\cdots$

$\cdots (\,) \, {\color{red}(} \, (\,) \, (\,) \, {\color{red}]} \,) \cdots$



Each factor is a factor of a Dyck (or well-matched) word.

Beyond sofic constraints: Motzkin-Dyck constraints

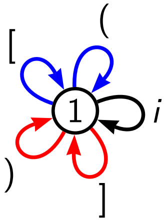
An allowed sequence:

$$\cdots [\, i\, (\, (i\, i\,)\,)\,]\, i\, [\, i\, i\, i\,]\, [\, (\cdots$$

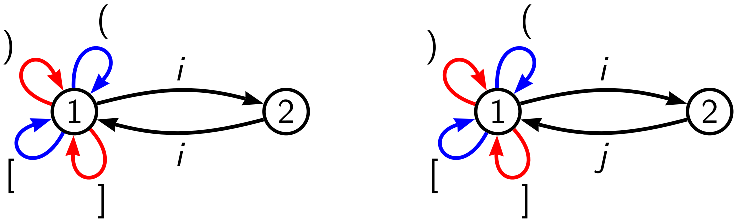
A forbidden sequence:

$$\cdots [\, i\, (\, (i\, i\,)\,)\,]\, i\, {\color{red}[}\, i\, i\, i\, {\color{red})}\, [\, (\cdots$$

The Motzkin constraint: the symbol "(" is matched with ")", the symbol "[" is matched with a "]".



Sofic-Dyck and finite-type-Dyck constraints



A sofic-Dyck constraint (left), a finite-type Dyck constraint (right)

The edge labeled by "(" (resp. "[") is matched with the edge labeled by ")" (resp. "]").

An allowed sequence for the sofic-Dyck constraint

$$\cdots [\, i \, (\, (i \, i \,) \,) \, i \,] \, [\, i \, i \, i \, i \,] \, [\, (\cdots$$

Generalize Markov-Dyck shifts (*Inoue, Krieger and Matsumoto*, 2010 and 2011).

Sofic-Dyck shifts

Shifts of sequences over a *pushdown alphabet* A which is the disjoint union of (A_c, A_r, A_i):
A_c is the set of call alphabet
A_r is the set of return alphabet
A_i is the set of internal alphabet

A Dyck automaton $(\mathcal{A}, M)$ over A is a directed labelled graph $\mathcal{A} = (Q, E, A)$ where $E \subset Q \times A \times Q$
M is the set of *matched edges*: a set of pairs $((p, a, q), (r, b, s))$ of edges of $\mathcal{A}$ with $a \in A_c$ and $b \in A_r$

equipped with a *graph semigroup* S generated by the set $E \cup \{x_{pq} \mid p, q \in Q\} \cup \{0\}$ with

Sofic-Dyck shifts

generators: $E \cup \{x_{pq} \mid p, q \in Q\} \cup \{0\}$

$$
\begin{aligned}
0s = s0 &= 0 && \text{for } s \in S, \\
x_{pq}x_{qr} &= x_{pr} && \text{for } p, q, r \in Q, \\
x_{pq}x_{rs} &= 0 && \text{for } p, q, r, s \in Q, q \neq r, \\
(p, \ell, q) &= x_{pq} && \text{for } p, q, \in Q, \ell \in A_i, \\
(p, a, q)x_{qr}(r, b, s) &= x_{ps} && \text{for } ((p, a, q), (r, b, s)) \in M, \\
(p, a, q)x_{qr}(r, b, s) &= 0 && \text{for } ((p, a, q), (r, b, s)) \notin M, \\
(p, a, q)(r, b, s) &= 0, && \text{for } p, q, r, s \in Q, q \neq r, a, b \in A, \\
x_{pp}(p, a, q) = (p, a, q) &= (p, a, q)x_{qq} && \text{for } p, q \in Q, a \in A, \\
x_{pq}(r, a, s) = 0 &= (r, a, s)x_{tu} && \text{for } p, q \in Q, a \in A, \ q \neq r, \ s \neq t.
\end{aligned}
$$

Sofic-Dyck shifts

If π is a finite path, $f(\pi)$ is its image in the graph semigroup S

- A finite path is admissible if $f(\pi) \neq 0$
- A word labeling an admissible path π such that $f(\pi) = x_{pq}$ is a Dyck word or a well-matched word
- A bi-infinite path is *admissible* if all its factors are admissible
- A bi-infinite sequence is *accepted* by $(\mathcal{A}, M)$ if it is the label of a bi-infinite admissible path of $(\mathcal{A}, M)$.

A *sofic-Dyck shift* is a set of bi-infinite sequences accepted by a Dyck automaton.

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$(p, \ell, q) \in \Delta \qquad p, \begin{array}{|c|} \\ \alpha \\ \vdots \\ \beta \\ \perp \\ \hline \end{array}$

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$$(p, \ell, q) \in \Delta \qquad p, \left|\begin{array}{c} \\ \alpha \\ \vdots \\ \beta \\ \perp \end{array}\right| \xrightarrow{\ \ell\ } q, \left|\begin{array}{c} \\ \alpha \\ \vdots \\ \beta \\ \perp \end{array}\right|$$

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$$(p, a, q, \alpha) \in \Delta \qquad p, \begin{array}{|c|} \\ \alpha \\ \vdots \\ \beta \\ \perp \\ \hline \end{array}$$

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$$(p, a, q, \alpha) \in \Delta \qquad p, \left|\begin{array}{c} \\ \alpha \\ \vdots \\ \beta \\ \perp \end{array}\right| \xrightarrow{a} q, \left|\begin{array}{c} \alpha \\ \alpha \\ \vdots \\ \beta \\ \perp \end{array}\right|$$

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$(p, b, \alpha, q) \in \Delta$ $\qquad p, \begin{array}{|c|} \\ \alpha \\ \vdots \\ \beta \\ \perp \\ \hline \end{array}$

Visibly pushdown automaton (Alur *et al.* 2004)

$M = (Q, I, \Gamma, \Delta, F)$

- Q is the finite state of states
- $A = (A_c, A_r, A_i)$ is the partitioned alphabet
- Γ is the stack alphabet
- $\Delta \subset \begin{cases} Q \times A_c \times Q \times (\Gamma \setminus \{\perp\}) \\ Q \times A_r \times (\Gamma \setminus \{\perp\}) \times Q \\ Q \times A_i \times Q \end{cases}$

$(p, b, \alpha, q) \in \Delta$

$$p, \begin{array}{|c|} \\ \alpha \\ \vdots \\ \beta \\ \perp \\ \hline \end{array} \xrightarrow{b} q, \begin{array}{|c|} \\ \\ \vdots \\ \beta \\ \perp \\ \hline \end{array}$$

Sofic-Dyck shifts = Visibly pushdown shifts

Proposition (B., Blockelet, Dima, preprint 2013)

The set of allowed blocks of a sofic-Dyck shift is a visibly pushdown language. Conversely, if L is a factorial extensible visibly pushdown language, then the shift of sequences whose factors belong to L is a sofic-Dyck shift.

It is not difficult to prove that the set of labels of finite admissible paths is a visibly pushdown language.

It is more complicate to prove that it holds also for the set of (allowed) blocks. Indeed, labels of finite admissible paths may not be blocks. Culik and Yu showed that the subset of bi-extensible words of a context-free language may not be context-free. It is true for factorial languages. We adapt the construction for the visibly pushdown case.

Zeta functions of shifts

Let (X, σ) be a shift where $\sigma : (x_i)_{i\in\mathbb{Z}} \to (x_{i+1})_{i\in\mathbb{Z}}$.
Denoting by p_n the number of sequences $x \in X$ such that $\sigma^n(x) = x$, the zeta function of X is defined as

$$\zeta_X(z) = \exp \sum_{n>0} \frac{p_n}{n} z^n = \prod_{\gamma \text{ periodic orbit}} (1 - z^{|\gamma|})^{-1}.$$

Periodic pattern *ababb*

$$\cdots ababb\ ababb\ ababb\ ababb \cdots$$

Formula of the zeta function

Shifts of finite type

Bowen and Lanford 1970

Sofic shifts

Manning 1971, Bowen 1978

Dyck shifts, *Keller* 1991

Motzkin shifts, *Inoue* 2006

Markov-Dyck shifts

Krieger and Matsumoto 2011

Formula of the zeta function

Shifts of finite type *Bowen and Lanford 1970*	Sofic shifts *Manning 1971, Bowen 1978*
Finite-type Dyck shifts *preprint 2013*	Sofic-Dyck shifts *preprint 2013*

Formula of the zeta function

Shifts of finite type $\mathbb{N}$-rational	Sofic shifts $\mathbb{N}$-rational, *Reutenauer 1997*

Formula of the zeta function

Shifts of finite type $\mathbb{N}$-rational	Sofic shifts $\mathbb{N}$-rational, *Reutenauer 1997*
Finite-type Dyck shifts $\mathbb{N}$-algebraic, *ITA2014*	Sofic-Dyck shifts $\mathbb{N}$-algebraic, *soon*

Zeta functions of sofic-Dyck shifts

Let $(\mathcal{A}, M)$ be a (deterministic, reduced) Dyck automaton.
We define the matrices

- $C = (C_{pq})$, where C_{pq} is the set of prime Dyck words labeling a path from p to q: well-matched words with no nonempty shorter well-matched prefix.
- $M_c = (M_{c,pq})$, (resp. M_r) where $M_{c,pq}$ is the sum of call (resp. return) letters labeling an edge from p to q).
- C_c (resp. C_r) is the matrix CM_c^* (resp. the matrix $M_r^* C$).
- $C(z) = (C_{pq}(z))$, where $C_{pq}(z)$ is the generating series of C_{pq}.

The matrices $X = C, C_c, C_r, M_c, M_r$, are circular:
$x_1 \in X_{p_0,p_1}$, $x_2 \in X_{p_1,p_2}$, $\ldots$,$x_n \in X_{p_{n-1}p_0}$, $y_1 \in X_{q_0,q_1}$, $y_2 \in X_{q_1,q_2}, \ldots, y_m \in X_{q_{m-1}q_0}$ and $p \in A^*$ and $s \in A^+$,

$$sx_2x_3 \cdots x_n p = y_1 y_2 \cdots y_m, \tag{1}$$

$$x_1 = ps \tag{2}$$

implies $n = m$, $p = \varepsilon$ and $x_i = y_i$.

Encoding of periodic sequences

Proposition (extending Krieger's result for Markov-Dyck shifts)

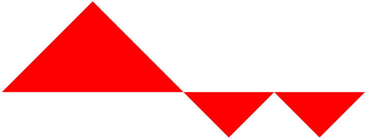
The zeta function of a sofic-Dyck shift accepted by a (deterministic, reduced) Dyck automaton $(\mathcal{A}, M)$ *with matrices* C, C_c, C_r, M_c, M_r *is given by the following formula.*

$$\zeta_X(z) = \frac{\zeta_{X_{C_c}}(z)\zeta_{X_{C_r}}(z)\zeta_{X_{M_c}}(z)\zeta_{X_{M_r}}(z)}{\zeta_{X_C}(z)}.$$

Encoding of periodic sequences. Case balance$(w) = 0$

a call symbol, b return symbol

$w = aabbbaba$

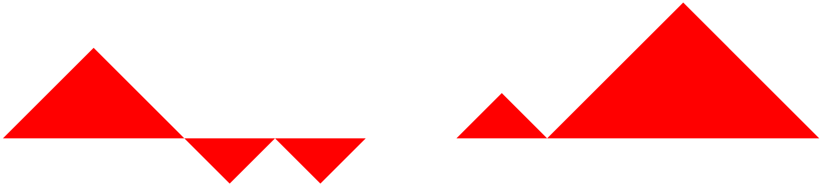


Encoding of periodic sequences. Case balance$(w) = 0$

a call symbol, b return symbol

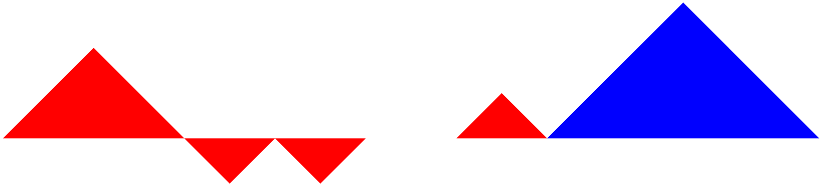
$w = aabbbaba$

$u = abaaabbb \in C^*$



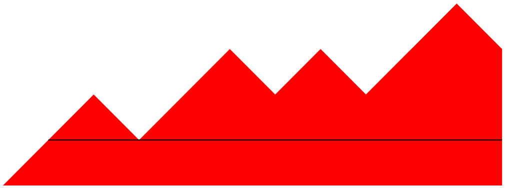
Encoding of periodic sequences. Case balance$(w) = 0$

a call symbol, b return symbol
$w = aabbbaba$
$u = abaaabbb \in C^*$



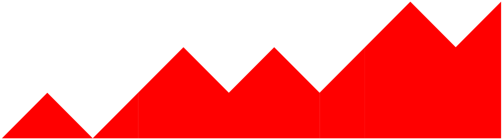
Encoding of periodic sequences. Case balance$(w) > 0$

$w = aabaababaaba$



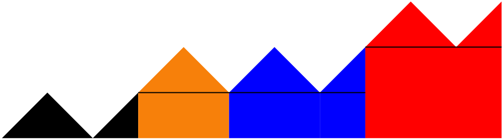
Encoding of periodic sequences. Case balance(w) > 0

$u = abaababaabaa$



Encoding of periodic sequences. Case balance$(w) > 0$

$u = abaababaabaa \in (CA_c{}^*)^*$



Zeta functions of sofic-Dyck shifts

Let $(\mathcal{A} = (Q, E), M)$ be a (deterministic, reduced) Dyck automaton.

$\mathcal{A}_{\otimes \ell}$ is the labelled graph with states $Q_{\otimes k}$, the set of all ordered k-uples of states of Q, and edges:

$$(p_1, \ldots, p_k) \xrightarrow{a} (q'_1, \ldots, q'_k)$$

if and only if

$$p_1 \xrightarrow{a} q_1$$
$$p_2 \xrightarrow{a} q_2$$
$$\ldots$$
$$p_k \xrightarrow{a} q_k$$

and $(q'_1, \ldots, q'_k)$ is an even permutation of $(q_1, \ldots, q_k)$.

Zeta functions of sofic-Dyck shifts

Let $(\mathcal{A} = (Q, E), M)$ be a (deterministic, reduced) Dyck automaton.

$\mathcal{A}_{\otimes \ell}$ is the labelled graph with states $Q_{\otimes k}$, the set of all ordered k-uples of states of Q, and edges:

$$(p_1, \dots, p_k) \xrightarrow{-a} (q'_1, \dots, q'_k)$$

if and only if

$$\begin{gathered} p_1 \xrightarrow{a} q_1 \\ p_2 \xrightarrow{a} q_2 \\ \dots \\ p_k \xrightarrow{a} q_k \end{gathered}$$

and $(q'_1, \dots, q'_k)$ is an odd permutation of $(q_1, \dots, q_k)$.

Zeta functions of sofic-Dyck shifts

Let $(\mathcal{A} = (Q, E), M)$ be a (deterministic, reduced) Dyck automaton.

$\mathcal{A}_{\otimes \ell}$ is the labelled graph with states $Q_{\otimes k}$, the set of all ordered k-uples of states of Q, and edges:

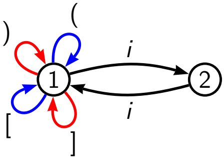
Proposition (B., Blockelet, Dima, preprint 2013)

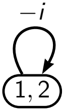
$$\zeta_X(z) = \prod_{\ell=1}^{|Q|} \det(I - C_{c,\otimes\ell}(z))^{(-1)^\ell} \det(I - C_{r,\otimes\ell}(z))^{(-1)^\ell}$$
$$\det(I - M_{c,\otimes\ell}(z))^{(-1)^\ell} \det(I - M_{r,\otimes\ell}(z))^{(-1)^\ell} \det(I - C_{\otimes\ell}(z))^{(-1)^\ell+1}.$$

Example

Let X accepted by $(\mathcal{A}, M)$ over $A = (\{(, [\}, \{),]\}, \{i\})$

Matched edges: $(1 \xrightarrow{(} 1,\ 1 \xrightarrow{)} 1)$,$(1 \xrightarrow{[} 1,\ 1 \xrightarrow{]} 1)$.





$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, = \begin{bmatrix} (\ D_{11}\ ) + [\ D_{11}\ ], & i \\ i & 0 \end{bmatrix}, C_{\otimes 2} = \left[C_{(1,2),(1,2)} \right] = [-i]$$

where $D_{11} = (\ D_{11}\ )\ D_{11} + [\ D_{11}\ ]\ D_{11} + i\ i\ D_{11} + \varepsilon$.

Example

$$C_c = C{M_c}^* = \begin{bmatrix} C_{11} & i \\ i & 0 \end{bmatrix} \begin{bmatrix} \{(,[\}^* & 0 \\ 0 & \varepsilon \end{bmatrix} = \begin{bmatrix} C_{11}\{(,[\}^* & i \\ i\{(,[\}^* & 0 \end{bmatrix},$$

$$C_r = {M_r}^* C = \begin{bmatrix} \{),]\}^* & 0 \\ 0 & \varepsilon \end{bmatrix} \begin{bmatrix} C_{11} & i \\ i & 0 \end{bmatrix} = \begin{bmatrix} \{),]\}^* C_{11} & \{),]\}^* i \\ i & 0 \end{bmatrix}.$$

$$\prod_{\ell=1}^{2} \det(I - M_{c,\otimes\ell}(z))^{(-1)^\ell} = \prod_{\ell=1}^{2} \det(I - M_{r,\otimes\ell}(z))^{(-1)^\ell} = \frac{1}{1-2z}.$$

Example

We finally get

$$\begin{aligned}\zeta_X(z) &= \frac{(1+z)(1-z^2-C_{11}(z))}{(1-2z-z^2-C_{11}(z))^2},\\ &= \frac{(1+z)(1-z^2-\frac{1-z^2-\sqrt{1-10z^2+z^4}}{2})}{(1-2z-z^2-\frac{1-z^2-\sqrt{1-10z^2+z^4}}{2})^2}.\end{aligned}$$

The entropy (or capacity) of the shift is

$$h(X) = \log\frac{1}{\rho} = \log\frac{2}{\sqrt{13}-3} \sim \log 3.3027.$$

$\mathbb{N}$-algebricity of zeta functions

Proposition

The zeta function of a sofic-Dyck shift is a computable $\mathbb{N}$-visibly pushdown series, i.e. is the (ordinary) generating series of some visibly pushdwon language.

Example continued

$$
\begin{aligned}
\zeta_X(z) &= \frac{(1+z)(1-z^2-C_{11}(z))}{(1-2z-z^2-C_{11}(z))^2}, \\
&= \frac{(1+z)(1-z^2-C_{11}(z))}{(1-2z-z^2-C_{11}(z))(1-z^2-C_{11}(z)-2z)}, \\
&= \frac{(1+z)}{(1-2z-z^2-C_{11}(z))(1-2z(z^2+C_{11}(z))^*)}, \\
&= (1+z)(2z+z^2+C_{11}(z))^*(2z(z^2+C_{11}(z))^*)^*.
\end{aligned}
$$

and $C_{11}(z)$ is the generating series of a VPL.

Sketch of proof

Let $(\mathcal{A}, M)$ be a (deterministic, reduced) Dyck automaton.

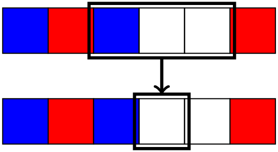
1. Use another encoding of periodic sequences.
   - (Krieger) M_c, M_r, $CM_c{}^*$, $M_r{}^*C$
   - Another encoding: C^*M_c, $M_r + C$
2. $\zeta_X(z) = \zeta_{X_{C^*M_c}}(z)\zeta_{X_{M_r+C}}(z)$.
3. Let M be a matrix with coefficients in A^+, $\mathcal{P}$ be a subset of the set of all pairs of states of $\mathcal{A}$ and $V_{M,\mathcal{P}}$ be the set of words belonging to M_{pq} if and only if $(p,q) \in \mathcal{P}$.
4. Let $B = \{a_{M,\mathcal{P}}\}$ be a set of new symbols.
5. Let $M = C^*M_c$ or $M = M_r + C$.
   If $M_{pq} = \sum_{\mathcal{P}|(p,q)\in\mathcal{P}} V_{M,\mathcal{P}}$ we set $N_{pq} = \sum_{\mathcal{P}|(p,q)\in\mathcal{P}} a_{M,\mathcal{P}}$.
   $$\zeta_{X_M}(z) = \theta Z(N)[a_{M,\mathcal{P}} \to \pi V_{M,\mathcal{P}}].$$
   where $Z(N)$ is the generalized zeta function of the set of labels of bi-infinite paths defined by N (Berstel, Reutenauer).
6. $Z(N)$ is $\mathbb{N}$-rational (Reutenauer) and all $V_{M,\mathcal{P}}$ are VPL. Thus $\zeta_{X_M}(z)$ is the generating function of a VPL language.

Finite-type-Dyck shifts

Finite-type-Dyck shift are accepted by *local* (or *definite*) Dyck automata.

We says that $(\mathcal{A}, M)$ is (m, a)-*local* if whenever two paths (or two admissible paths) $(p_i, a_i, p_{i+1})_{-m \leq i \leq a}$, $(q_i, a_i, q_{i+1})_{-m \leq i \leq a}$, of $\mathcal{A}$ of length $m + a$ have the same label, then $p_0 = q_0$.

Proper block map



A map $\Phi : X \to Y$ is called an (m, a)-*local map* (or an (m, a)-*block map*) if there exists a function $\phi : \mathcal{B}_{m+a+1}(X) \to B$ such that $\Phi(x)_i = \phi(x_{i-m} \cdots x_{i-1} x_i x_{i+1} \cdots x_{i+a})$.

A block map $\Phi : \mathsf{X}_A \to \mathsf{X}_{A'}$, where $A = (A_c, A_r, A_i)$ and $A' = (A'_c, A'_r, A'_i)$, is *proper* if $\Phi(x)_j \in A'_c$ (resp. A'_r, A'_i) whenever $x_j \in A_c$ (resp. A_r, A_i) for any j.

Proper conjugacy: conjugacy which is a proper block map.

Finite-type-Dyck shifts

Proposition

A subshift is a sofic-Dyck shift if and only it is the proper factor of a finite-type-Dyck shift.

Corollary

A proper factor of a sofic-Dyck shift is a sofic-Dyck shift.

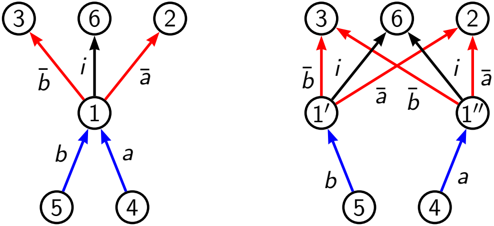
In-split of a Dyck automaton

$(\mathcal{A} = (Q, E, A), M)$ over $A = (A_c, A_r, A_i)$
Let $p \in Q$ and $\mathcal{P}$ a partition $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ of the edges coming in p.
$(\mathcal{A}' = (Q', E', A), M')$ is defined by

- $Q' = Q \setminus \{p\} \cup \{p_1, \ldots, p_k\}$,
- $(q, a, r) \in E'$ if $q, r \neq p$ and $(p, a, r) \in E$,
- $(q, a, p_i) \in E'$ for each i such that $(q, a, p) \in \mathcal{P}_i$,
- $(p_i, a, r) \in E'$ for each i such that $(p, a, r) \in E$.
- M' is the set of pairs of edges $(q, a, r), (s, b, t)$ where $a \in A_r, b \in A_c$ such that $(\pi(q), a, \pi(r)), (\pi(s), b, \pi(t)) \in M$ where $\pi(q) = q$ for $q \neq p$ and $\pi(p_i) = p$.

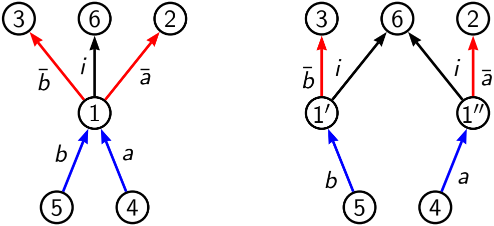
Example

A Dyck state-splitting of the state 1 into $1'$ and 1".



Trim in-split of a Dyck automaton

A trim Dyck state-splitting of the state 1 into $1'$ and 1".
Edges (p_i, a, r) which are not essential in $(\mathcal{A}', M')$ are removed from E'.
Matched pairs $(q, a, r), (p_i, b, t)$ or $(p_i, b, t), (q, a, r)$ which are not essential are removed from M'.



Edge-Dyck shifts

A *Dyck graph* $(\mathcal{G} = (Q, E \subset Q \times Q), M)$ is composed of a graph $\mathcal{G}$, where the edges $E = (E_c, E_r, E_i)$ are partitioned into three categories (*call edges*, *return edges*, and *internal edges*).

An *edge-Dyck shift* $\mathsf{X}_{(\mathcal{G},M)}$ is the set of admissible bi-infinite paths of a Dyck graph.

Proposition

Each finite-type-Dyck shift is properly conjugate to a finite-type edge-Dyck shift.

A decomposition theorem for edge-Dyck shifts

Theorem

Let (G, M), $(\mathcal{H}, N)$ be two Dyck graphs such that $X_{(\mathcal{G},M)}$ and $X_{(\mathcal{H},N)}$ are properly conjugate. Then there are finite sequences of Dyck graphs $(\mathcal{G}_i, M_i)$, $(\mathcal{H}_j, N_j)$ and Dyck (or trim Dyck) in-splittings $\Psi_i : (\mathcal{G}_i, M_i) \to (\mathcal{G}_{i+1}, M_{i+1})$, $\Delta_j : (\mathcal{H}_j, N_j) \to (\mathcal{H}_{j+1}, N_{j+1})$, such that $(\mathcal{G}_1, M_1) = (\mathcal{G}, M)$, $(\mathcal{H}_1, N_1) = (\mathcal{H}, N)$, and $(\mathcal{G}_k, M_k) = (\mathcal{H}_{k'}, N_{k'})$, up to renaming of the states.

$$(\mathcal{G}, M) \xrightarrow{\Psi_1} \ldots \xrightarrow{\Psi_k} (\mathcal{G}_k, M_k) = (\mathcal{H}_{k'}, N_{k'}) \xleftarrow{\Delta_{k'}} \ldots \xleftarrow{\Delta_1} (\mathcal{H}, N)$$

In-splittings commute but (unfortunately) not trim in-splittings.