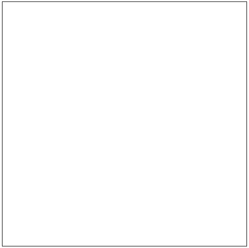
Nearest Neighbour shifts of finite type

Nishant Chandgotia

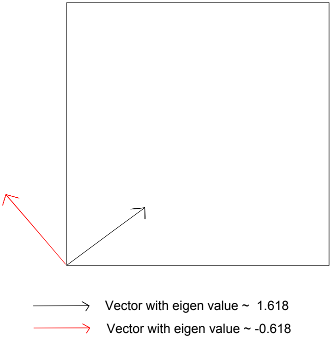
University of British Columbia

November 2013

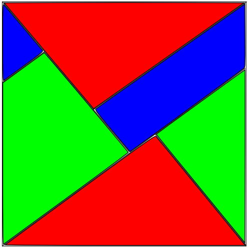
Consider a torus $\mathbb{R}^2/\mathbb{Z}^2$ with the map $\left[\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right]$.



$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ has two eigenvalues (~ 1.618 and $-.618$).



We can divide the torus into 3 parts by extending the eigendirections. These are called Markov partitions.



Trajectories of points of the torus under the map $\left[\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ can be coded via the partition it visits.

Trajectories of points of the torus under the map $\left[\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right]$ can be coded via the partition it visits. This gives us an 'isomorphism' between the torus with the map $\left[\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right]$ and the set of bi-infinite sequences in red, green and blue where consecutive colours cannot be the same and red cannot be followed by green.

Trajectories of points of the torus under the map $\left[\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right]$ can be coded via the partition it visits. This gives us an 'isomorphism' between the torus with the map $\left[\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right]$ and the set of bi-infinite sequences in red, green and blue where consecutive colours cannot be the same and red cannot be followed by green.

This phenomenon is much more general: Any automorphism of the torus(with no eigenvalues of modulus 1) can be coded in a similar way.

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set.

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set. A shape F is any finite subset of $\mathbb{Z}^d$.

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set. A shape F is any finite subset of $\mathbb{Z}^d$. A pattern is a function $f : F \longrightarrow \mathcal{A}$.

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set. A shape F is any finite subset of $\mathbb{Z}^d$. A pattern is a function $f : F \longrightarrow \mathcal{A}$. Given $\mathcal{F}$, a list of patterns,

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set. A shape F is any finite subset of $\mathbb{Z}^d$. A pattern is a function $f : F \longrightarrow \mathcal{A}$. Given $\mathcal{F}$, a list of patterns, a shift space $X_{\mathcal{F}}$ is given by

$$X_{\mathcal{F}} = \{x \in \mathcal{A}^{\mathbb{Z}^d} \mid \text{ patterns from } \mathcal{F} \text{ do not occur in } x\}.$$

Nearest Neighbour Shifts of Finite Type

Let $\mathcal{A}$ be a finite alphabet set. A shape F is any finite subset of $\mathbb{Z}^d$. A pattern is a function $f : F \longrightarrow \mathcal{A}$. Given $\mathcal{F}$, a list of patterns, a shift space $X_{\mathcal{F}}$ is given by

$$X_{\mathcal{F}} = \{x \in \mathcal{A}^{\mathbb{Z}^d} \mid \text{ patterns from } \mathcal{F} \text{ do not occur in } x\}.$$

A nearest neighbour shift of finite type is a shift space such that $\mathcal{F}$ can be given by patterns on ‘edges’.

Examples:

- The full shift: $\mathcal{F}$ is empty. $X_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}^d}$.

Examples:

- The full shift: $\mathcal{F}$ is empty. $X_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}^d}$.
- The hard square model: $\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11\}_{i=1}^{d}$.

Examples:

- The full shift: $\mathcal{F}$ is empty. $X_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}^d}$.
- The hard square model: $\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11\}_{i=1}^{d}$.
- The n-coloured chessboard: $\mathcal{A} = \{0, 1, 2, \ldots, n-1\}$ and $\mathcal{F} = \{00, 11, 22, \ldots,\}_{i=1}^{d}$.

Examples:

- The full shift: $\mathcal{F}$ is empty. $X_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}^d}$.
- The hard square model: $\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11\}_{i=1}^{d}$.
- The n-coloured chessboard: $\mathcal{A} = \{0, 1, 2, \ldots, n-1\}$ and $\mathcal{F} = \{00, 11, 22, \ldots, \}_{i=1}^{d}$.

0	1	0	1	0	1	0	1
1	2	1	0	1	2	1	0
0	1	0	1	0	1	0	1
1	0	1	2	1	0	1	2
0	1	0	1	0	1	0	1
1	0	1	2	1	2	1	0
0	1	0	1	0	1	0	1
1	2	1	0	1	2	1	0

Figure: The 3-coloured chessboard in 2 dimensions

1 Dimension vs Higher Dimensions

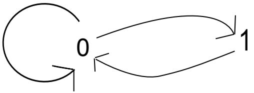
A lot is known about 1 dimensional nearest neighbour shifts of finite type

1 Dimension vs Higher Dimensions

A lot is known about 1 dimensional nearest neighbour shifts of finite type because they can be recoded as walks on graphs (vertex shifts).

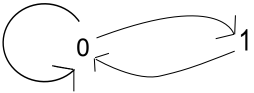
1 Dimension vs Higher Dimensions

A lot is known about 1 dimensional nearest neighbour shifts of finite type because they can be recoded as walks on graphs (vertex shifts).For instance walks on



1 Dimension vs Higher Dimensions

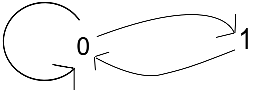
A lot is known about 1 dimensional nearest neighbour shifts of finite type because they can be recoded as walks on graphs (vertex shifts).For instance walks on



give us the hard square shift space. ($\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11\}$).

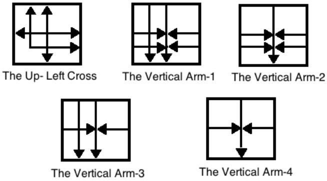
1 Dimension vs Higher Dimensions

A lot is known about 1 dimensional nearest neighbour shifts of finite type because they can be recoded as walks on graphs (vertex shifts).For instance walks on



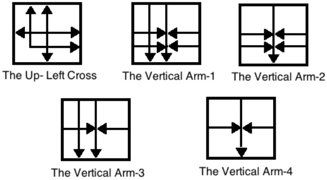
give us the hard square shift space. ($\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11\}$). However in higher dimensions given $\mathcal{A}$ and $\mathcal{F}$ there is no algorithm to decide whether the nearest neighbour shift of finite type is non-empty!!!

Let the alphabet be



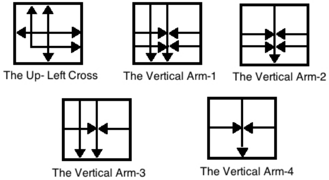
and their rotations.

Let the alphabet be

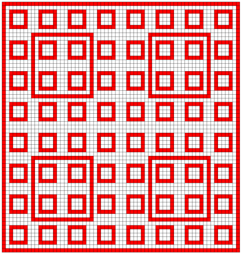


and their rotations. The adjacency rules entail that the arrows should match up.

Let the alphabet be



and their rotations. The adjacency rules entail that the arrows should match up. The double arrows form patterns which look like



.

It turns out that all elements of the shift space are non-periodic

It turns out that all elements of the shift space are non-periodic and given a turing machine one can mimic its working with a slight modification of the symbols described above.

It turns out that all elements of the shift space are non-periodic and given a turing machine one can mimic its working with a slight modification of the symbols described above.
These shift spaces are non-empty if and only if the corresponding turing machine does not halt on an empty input.

It turns out that all elements of the shift space are non-periodic and given a turing machine one can mimic its working with a slight modification of the symbols described above.
These shift spaces are non-empty if and only if the corresponding turing machine does not halt on an empty input.Since the latter is undecidable it is undecidable whether or not given an alphabet and its adjacency rules, the shift space generated is non-empty.

Thank You!

.