

AN ABSTRACT OF THE DISSERTATION OF

Larry A. Pierce II for the degree of Doctor of Philosophy in Mathematics presented on July 23, 2008.

Title: Computing Entropy for \mathbb{Z}^2 -actions

Abstract Approved:

Robert M. Burton

Abstract: For a certain class of \mathbb{Z}^2 -actions, we provide a proof of a conjecture that the ratio of the Perron eigenvalues of the transfer matrices of the free boundary restrictions converge to the entropy of that action. Also, a novel method for computing the entropy of \mathbb{Z}^2 -actions is conjectured.

Computing Entropy for \mathbb{Z}^2 -actions

by
Larry A. Pierce II

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented July 23, 2008

Commencement June 2009

Doctor of Philosophy dissertation of Larry A. Pierce II presented
on July 23, 2008

APPROVED:

Major Professor, representing Mathematics

Chair of the Department of Mathematics

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Larry A. Pierce II, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor, Bob Burton, for his countless hours of input and contributions to this work.

I would also like to thank Paul Shields for his guidance and knowledge in the pursuit of this work.

I dedicate this work to my wife, Mariana Schatte, whose endless support has made this dissertation possible.

TABLE OF CONTENTS

Contents

0	Outline	1
1	An introduction to shifts of finite type	2
2	An introduction to \mathbb{Z}^2-actions	10
2.1	Examples of \mathbb{Z}^2 -actions	14
3	Generating the transition matrices for \mathbb{Z}^2-actions	18
4	A reflection operator for \mathbb{Z}^2-actions	26
5	Computing entropy for \mathbb{Z}^2-actions	32
6	Asymptotic Analysis of the Substitution Algorithm and its Relationship to Shifts of Finite Type	45
7	Geometry of the V_n matrices and their eigenvectors	59
8	Algebraic manipulations of the substitution algorithm	70
9	Numerical Estimations for the Entropy of \mathbb{Z}^2-actions	79
10	Bibliography	88
A	Matlab Code	90

Computing Entropy for \mathbb{Z}^2 -actions

0 Outline

Discussed in this dissertation are methods for computing the entropy for a class of \mathbb{Z}^2 -actions. Chapter 1 gives an introduction to one dimensional shift systems and the concept of entropy for those systems. The standard method for computing entropy for such systems is outlined. Chapter 2 is a discussion of extending shift systems to two dimensional \mathbb{Z}^2 -actions. In chapter 3, a general method is introduced for generating the sequence of transfer matrices for the free boundary restrictions of these \mathbb{Z}^2 -actions. Chapter 4 defines and discusses the properties of a reflection operator which is used in the following chapters. Chapter 5 describes a long-standing conjecture about \mathbb{Z}^2 -actions, and proves the conjecture for a specific class of \mathbb{Z}^2 -actions. Chapters 6, 7, and 8 deal with various types of analysis and manipulations of the algorithm presented in chapter 3. Chapter 9 proposes a new type of numerical estimation for entropy with examples of its power and limitations.

1 An introduction to shifts of finite type

We begin with a short introduction to *shift systems* and *shifts of finite type*. A full treatment of this material can be found in [15] and [12].

Let \mathbb{Z} denote the set of integers and \mathbb{Z}^+ denote the set of positive integers. We consider a set, \mathcal{A} , called the **alphabet**, and the elements of \mathcal{A} will be known as **characters**, or **letters**. We will adopt the notation $\mathcal{A} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots, \mathbf{N}\}$, reserving the different fonts for alphabets and letters. An element of \mathcal{A}^n will be known as a **word**, or a **block** of length n . If $\alpha \in \mathcal{A}^{\mathbb{Z}}$ (or if $\alpha \in \mathcal{A}^{\mathbb{Z}^+}$), we will denote by α_i the element in \mathcal{A} which is the projection of α onto the i^{th} coordinate. For $i < j$, by α_i^j we mean the word of length $j - i + 1$ which is the projection of α onto the coordinates i, \dots, j .

In other words, if $\alpha \in \mathcal{A}^{\mathbb{Z}}$:

$$\alpha = \dots \alpha_{-2} \alpha_{-1} \alpha_0 \alpha_1 \alpha_2 \dots$$

And:

$$\alpha_{-1}^2 = \alpha_{-1} \alpha_0 \alpha_1 \alpha_2 \in \mathcal{A}^4$$

We also adopt the convention $\mathcal{A}^* = \bigcup_{n=0}^{\infty} \mathcal{A}^n$, the set of all finite words with letters taken from \mathcal{A} .

We consider a set $\mathcal{F} \subset \mathcal{A}^*$, which we will call the **forbidden set**.

We define the **shift operator** $\sigma : \mathcal{A}^{\mathbb{Z}} \rightarrow \mathcal{A}^{\mathbb{Z}}$ to be the map which “shifts” points of \mathcal{A} one unit “to the left”. That is to say: $\forall \alpha \in \mathcal{A}^{\mathbb{Z}}, \sigma(\alpha)_n = \alpha_{n+1}$.

We call $\mathcal{G} \subset \mathcal{A}^{\mathbb{Z}}$ a **shift system** with forbidden set \mathcal{F} if for every $\alpha \in \mathcal{G}$, $\alpha_i^j \notin \mathcal{F} \quad \forall i, j \in \mathbb{Z}$. Note that if \mathcal{G} is a shift system, then $\alpha \in \mathcal{G} \Rightarrow \sigma(\alpha) \in \mathcal{G}$.

Note that throughout this paper, \mathcal{G} will be used to denote any shift system.

The reader should make note of which system \mathcal{G} represents according to context. (Similarly for \mathcal{F} .)

At this point, we examine an example of a shift system which will be often referred to, and later generalized in higher dimensions.

Example 1.1 *The **Fibonacci shift** is defined as the shift system which has alphabet $\mathcal{A} = \{0, 1\}$ and forbidden language $\mathcal{F} = \{11\}$. That is to say, the system of all bi-infinite binary sequences in which no two 1's appear consecutively.*

Now, there is an important subtlety to note. In the Fibonacci shift we could have defined the forbidden set to be $\mathcal{F} = \{11, 111, 1111, \dots\}$, or $\mathcal{F} = \{11, 110, 1100, 11000, \dots\}$, or any other set which achieves the same result. The fact that we can express the forbidden blocks for this system by a finite set turns out to be extremely important.

There are shift systems which do not have this property. The following example is an illustration.

Example 1.2 *We define the **even shift** to have a binary alphabet, $\mathcal{A} = \{0, 1\}$ in which between any two 1's, there are an even number of zeros. For example,*

1001001000010000001001 would be an allowed block in the even shift.

However, 10010001001 would not be an allowed block because an odd number of zeros appear between consecutive ones. It is easy to see that the even shift cannot be expressed by any finite forbidden set. However, it is still an invariant set under the shift operator and is thus still a shift system.

■

This prompts us to make a stipulative distinction between shift systems as the Fibonacci and Even shifts.

Definition 1.1 We define a *shift of finite type* to be a shift system that can be expressed with a finite forbidden set.

With our current examples, we see that the Fibonacci shift is a shift of finite type, while the even shift is not.

A very important notion in dealing with shift systems is to know how many possible words of length n are allowed for any given system, as well as the rate at which that number increases with n . At this point we make another definition. For a given shift system, \mathcal{G} , we will denote by $\mathcal{B}_n(\mathcal{G})$ the set of allowed blocks of length n in \mathcal{G} .

For example, if \mathcal{G} is the Fibonacci system described above, then $\mathcal{B}_3(\mathcal{G}) = \{000, 001, 010, 100, 101\}$. In fact, it is easy to see that

$|\mathcal{B}_n(\mathcal{G})| = f_{n+2}$, where f_n is the n^{th} Fibonacci number (where the first six Fibonacci numbers are 1,1,2,3,5,8, with $f_6 = 8$).

However, it is easily realized that there are more complicated shift systems in which counting the number of n -blocks can be difficult for arbitrary n . The machinery of linear algebra gives us a powerful tool with which to compute these numbers.

Consider now the **full k -shift**, which consists of all bi-infinite sequences taken from an alphabet consisting of k letters. Thus, $\mathcal{F} = \{\emptyset\}$. Obviously,

$|\mathcal{B}_n(\mathcal{G})| = k^n$. In particular, for the full 2-shift, the number of allowed n -blocks is 2^n . Thus, for any shift on a binary alphabet, the number of allowed n -blocks must be less than or equal to 2^n , since the addition of a forbidden set can only reduce the number of allowable sequences. However, if the forbidden set is not too restrictive, we still expect the number of n -blocks to grow exponentially. It is this growth rate which is our focus.

For an arbitrary shift system on a binary alphabet, we expect the number of n -blocks to grow like $|\mathcal{B}_n(\mathcal{G})| \sim 2^{hn}$, where h is a number between zero and one. The number h is of particular interest, as it will yield a growth rate for the system.

Definition 1.2 *define the **entropy** for a shift system \mathcal{G} to be:*

$$h(\mathcal{G}) := \lim_{n \rightarrow \infty} \frac{1}{n} \log |\mathcal{B}_n(\mathcal{G})|$$

Where the logarithm is base 2. **Note now that any logarithm in this paper will be considered to be base 2.** We use the logarithm base 2, regardless of the size of the alphabet.

First, we need to show that this limit exists. Note that within a given shift system, we cannot always concatenate an m -block with an n -block to get an allowed word. This immediately tells us that:

$$|\mathcal{B}_{m+n}(\mathcal{G})| \leq |\mathcal{B}_m(\mathcal{G})| \times |\mathcal{B}_n(\mathcal{G})|$$

$$\implies \log |\mathcal{B}_{m+n}(\mathcal{G})| \leq \log |\mathcal{B}_m(\mathcal{G})| + \log |\mathcal{B}_n(\mathcal{G})|$$

Thus, the sequence $\{\log |\mathcal{B}_n(\mathcal{G})|\}$ is a subadditive sequence. It is known that if $\{a_n\}$ is a subadditive sequence, then the sequence a_n/n converges, and $a_n/n = \sup_n a_n/n$ (See [15]). Thus, the subadditivity of the logarithms of the number of n -blocks ensures that the entropy limit exists. Computing the entropy of a shift system is another challenge.

We now look at the *graph* of a shift system. Consider the following directed graph:

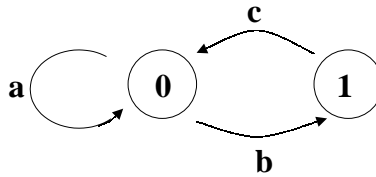


Figure 1: This graph represents the Fibonacci shift.

If we consider bi-infinite walks on the graph in Figure 1, and record each vertex visited and the time at which it was visited, we obtain a bi-infinite binary sequence. Also note that in this sequence, no two 1's will occur consecutively. That is to say that this graph perfectly represents the Fibonacci shift system.

It is easily seen that if \mathcal{G} is a shift of finite type, then there exists a graph which can be used to represent \mathcal{G} ([15]). However, some shift systems which are not of finite type do have a graph representation (the even shift has a graph representation). There are other shifts (not of finite type) which do not have finite graph representations.

The graph in Figure 1 has adjacency matrix $\hat{T} = \begin{bmatrix} a & b \\ c & 0 \end{bmatrix}$.

We see that: $\hat{T}^2 = \begin{bmatrix} aa + bc & ab \\ ca & bc \end{bmatrix}$, $\hat{T}^3 = \begin{bmatrix} aaa + abc + bca & aab + bcb \\ caa + cbc & cab \end{bmatrix}$

As is well-known (see [15]), the powers of \hat{T} spell out the paths generated by the graph. If we simply express \hat{T} as $T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, we can then count the number of paths, without actually knowing what those paths are.

Taking T to higher powers then allows us to know the number of allowed words of length n . That is to say that the number of allowed n -blocks for the Fibonacci shift, \mathcal{G} , is given by the sum of all entries of the adjacency matrix to the $n - 1$ power:

$$|\mathcal{B}_n(\mathcal{G})| = \sum_{i,j} [T^{n-1}]_{i,j} \quad (1)$$

Thus by knowing the growth rate of the entries of adjacency matrix, T , we can compute the entropy for the system \mathcal{G} . The Perron-Frobenius theorem ensures that if T is irreducible, then it has a positive eigenvalue, λ_T , which has multiplicity one and is the largest eigenvalue in absolute value. Furthermore, there is a corresponding eigenvector which has only non-negative entries. That is, if γ is any eigenvalue for T , then $|\gamma| \leq \lambda_T$. We call λ_T the **Perron eigenvalue** for T , and its corresponding eigenvector with norm 1 and all positive entries, \vec{v}_T , is known as the **Perron eigenvector**.

Let T be an $n \times n$ irreducible matrix, λ_T and $\vec{v} = \vec{v}_T$ be the Perron eigenvalue and Perron eigenvector for T , $a = \min_i [\vec{v}]_i$ and $b = \max_i [\vec{v}]_i$. Note that

$$T^k \vec{v} = \lambda_T^k \vec{v}.$$

Thus,

$$\sum_{j=1}^n [T^k]_{i,j} a \leq \sum_{j=1}^n [T^k]_{i,j} \vec{v}_j = \lambda_T^k \vec{v}_i \leq \lambda_T^k b \quad \forall i \in \{1, 2, \dots, n\}$$

which yields

$$\sum_{i=1}^n \sum_{j=1}^n [T^k]_{i,j} \leq \sum_{i=1}^n \left(\frac{b}{a}\right) \lambda_T^k = \left(\frac{nb}{a}\right) \lambda_T^k.$$

Also note that

$$a \lambda_T^k \leq \lambda_T^k \vec{v}_i = \sum_{j=1}^n [T^k]_{i,j} \vec{v}_j \leq \sum_{j=1}^n [T^k]_{i,j} b \leq \sum_{i=1}^n \sum_{j=1}^n [T^k]_{i,j} b$$

and therefore

$$\left(\frac{a}{b}\right) \lambda_T^k \leq \sum_{i=1}^n \sum_{j=1}^n [T^k]_{i,j}.$$

Thus giving us

$$\left(\frac{a}{b}\right) \lambda_T^k \leq \sum_{i,j=1}^n [T^k]_{i,j} \leq \left(\frac{nb}{a}\right) \lambda_T^k.$$

Equation 1 tells us that the term in the center is the number of words of length $k + 1$. Taking the logarithm and dividing by k yields

$$h(\mathcal{G}) = \log \lambda_T.$$

This equation allows us to easily compute the entropy for any shift of finite type by simply determining the eigenvalue of the transfer matrix for that system. For shifts of finite type, we are thus guaranteed to easily compute entropy.

For the Fibonacci shift, $\lambda_T = \frac{1+\sqrt{5}}{2}$, the "*golden ratio*". Thus, the entropy for the Fibonacci shift is $h(\mathcal{G}) = \log\left(\frac{1+\sqrt{5}}{2}\right)$.

2 An introduction to \mathbb{Z}^2 -actions

We now turn our focus to a different type of shift system. In the previous section, we looked at maps from the set of integers to a finite alphabet. An element of such a system was realized as an infinite string of characters taken from the alphabet. In this section, we extend this notion to two dimensions by looking at maps from the integer lattice, \mathbb{Z}^2 to a finite alphabet. Thus, a point in our space will be an element of $\mathcal{A}^{\mathbb{Z}^2}$.

In this setting, the forbidden language will be a set of labeled rectangles, rather than one dimensional strings.

Like the previous section, for $\alpha \in \mathcal{A}^{\mathbb{Z}^2}$, we will define $\alpha_{(i,j)} \in \mathcal{A}$ to be the projection of α onto its (i, j) coordinate. Also, for $r \geq i$ and $s \geq j$, we will define $\alpha_{(i,j)}^{(r,s)}$ to be the “**rectangle**” which consists of vertices labeled by letters in \mathcal{A} and agrees with those coordinates of α . For example, a typical α will look like:

$$\begin{array}{c} \vdots \\ \dots \quad \begin{array}{|c|c|c|c|c|} \hline \alpha_{(-2,2)} & \alpha_{(-1,2)} & \alpha_{(0,2)} & \alpha_{(1,2)} & \alpha_{(2,2)} \\ \hline \alpha_{(-2,1)} & \alpha_{(-1,1)} & \alpha_{(0,1)} & \alpha_{(1,1)} & \alpha_{(2,1)} \\ \hline \alpha_{(-2,0)} & \alpha_{(-1,0)} & \alpha_{(0,0)} & \alpha_{(1,0)} & \alpha_{(2,0)} \\ \hline \alpha_{(-2,-1)} & \alpha_{(-1,-1)} & \alpha_{(0,-1)} & \alpha_{(1,-1)} & \alpha_{(2,-1)} \\ \hline \alpha_{(-2,-2)} & \alpha_{(-1,-2)} & \alpha_{(0,-2)} & \alpha_{(1,-2)} & \alpha_{(2,-2)} \\ \hline \end{array} \quad \dots \\ \vdots \end{array}$$

Which represents a “**shift of finite type**” of the integer lattice with each vertex assuming a letter from \mathcal{A} . Then $\alpha_{(-2,-1)}^{(1,1)}$ will be the 3×4 rectangle:

$\alpha_{(-2,1)}$	$\alpha_{(-1,1)}$	$\alpha_{(0,1)}$	$\alpha_{(1,1)}$
$\alpha_{(-2,0)}$	$\alpha_{(-1,0)}$	$\alpha_{(0,0)}$	$\alpha_{(1,0)}$
$\alpha_{(-2,-1)}$	$\alpha_{(-1,-1)}$	$\alpha_{(0,-1)}$	$\alpha_{(1,-1)}$

Note that we are using tiled boxes to represent vertices on the integer lattice.

Similar to section 1, we also define

$$\mathcal{A}^* := \emptyset \cup \left\{ \bigcup_{i,j} \mathcal{A}_{(0,0)}^{(i,j)} \right\}$$

where i, j range over the non-negative integers.

This is the union of all rectangular restrictions of \mathbb{Z}^2 with corners $(0, 0)$ and (i, j) , and with each coordinate of that rectangle being assigned a letter from \mathcal{A} . As in section 1, we include the empty word, \emptyset .

This symbolic system is now indexed by the 2-lattice group, \mathbb{Z}^2 . We can define two shift operators, the vertical shift σ_v , and the horizontal shift σ_h , with the properties $\forall i, j \in \mathbb{Z}$, $[\sigma_v(\alpha)]_{(i,j)} = \alpha_{(i,j+1)}$, and $\forall i, j \in \mathbb{Z}$, $[\sigma_h(\alpha)]_{(i,j)} = \alpha_{(i+1,j)}$. We can think of σ_h as shifting α one unit to the left, and σ_v as shifting α one unit down.

As in section 1, we can define a forbidden set $\mathcal{F} \subset \mathcal{A}^*$.

Definition 2.1 For a finite alphabet set, \mathcal{A} , we call $\mathcal{G} \subset \mathcal{A}^{\mathbb{Z}^2}$ a **two-dimensional shift system** or **\mathbb{Z}^2 -action** with forbidden set $\mathcal{F} \subset \mathcal{A}^*$ if $\forall \alpha \in \mathcal{G}$ and $\forall (i, j), (r, s) \in \mathbb{Z}^2$, we have that $\alpha_{(i,j)}^{(r,s)} \notin \mathcal{F}$.

Definition 2.2 We say that a \mathbb{Z}^2 -action, \mathcal{G} , is of **finite type** if \mathcal{G} can be expressed by using a finite forbidden set, \mathcal{F} , composed of blocks of finite size.

We immediately see that if $\alpha \in \mathcal{G}$, then $\sigma_v(\alpha)$ and $\sigma_h(\alpha)$ are both in \mathcal{G} , hence, \mathcal{G} is shift-invariant under \mathbb{Z}^2 , the free abelian group on the two generators $\{\sigma_v, \sigma_h\}$.

We define $\mathcal{B}_{m,n}(\mathcal{G})$ to be the set of all allowed $m \times n$ blocks in \mathcal{G} , similar to section 1.

Example 2.1 (Hard Squares) We will use as our main example the **Fibonacci \mathbb{Z}^2 -action** (also known as the **Hard Square system**), in which each vertex of \mathbb{Z}^2 is labeled with a letter from the alphabet $\mathcal{A} = \{0, 1\}$ and with forbidden set:

$$\mathcal{F} = \left\{ \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} , \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array} \right\}$$

Note again that vertices on the integer lattice are represented by squares.

Then we have that all allowed 2×2 blocks are given by:

$$\mathcal{B}_{2,2}(\mathcal{G}) = \left\{ \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \right\}$$

Simply described, no two 1's can share an edge, but they can share a vertex diagonally.



Again, we are interested in how many allowed $n \times n$ blocks are allowed for a given \mathbb{Z}^2 -action, \mathcal{G} , and the growth rate of that sequence. However, we expect now that $|\mathcal{B}_{n,n}(\mathcal{G})| \sim 2^{hn^2}$.

Definition 2.3 We define the *entropy* of a \mathbb{Z}^2 -action to be

$$h(\mathcal{G}) := \lim_{n \rightarrow \infty} \frac{1}{n^2} \log |\mathcal{B}_{n,n}(\mathcal{G})|.$$

This limit exists by a subadditivity argument analogous and equivalent to the one given earlier for the one dimensional case.

For the one-dimensional case, we had an easy way to compute entropy for shifts of finite type. It turns out that in the two dimensional case no such machinery currently exists. Entropy has been computed exactly for only a few \mathbb{Z}^2 -actions, and most of those solutions have either been trivial or used very clever ideas which cannot be adapted to compute entropy for other systems. (See [14] and [5].)

It is the goal of the remainder of this paper to introduce methods which can be easily generalized to compute entropy for any \mathbb{Z}^2 -action. There is currently no general method to compute the exact entropy for any \mathbb{Z}^2 -action. However, we will explore numerical approximations for entropy, and we will examine those which give the greatest accuracy with the least of computing time.

2.1 Examples of \mathbb{Z}^2 -actions

We now pause to define and make some observations about several other examples of \mathbb{Z}^2 -actions which are used throughout this paper.

Example 2.2 (Vertically Independent Fibonacci Strips) *We will define the **Vertically Independent Fibonacci \mathbb{Z}^2 -action**, \mathcal{S}_h , to be the \mathbb{Z}^2 -action which consists of stacks of horizontal one-dimensional Fibonacci shifts with no vertical correlations, a property which we will refer to as **vertical independence**. More precisely, a **vertically independent system** is one in which the forbidden set consists of only $1 \times n$ blocks.*

We see that \mathcal{S}_h is the \mathbb{Z}^2 -action with alphabet

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

and forbidden set

$$\mathcal{F} = \{ \boxed{1} \boxed{1} \} .$$

■

Example 2.3 (Horizontally Independent Fibonacci Strips) *We will also make use of the system which consists of infinite vertical strips of Fibonacci shifts, with no horizontal correlations. We define a \mathbb{Z}^2 -action to be **horizontally independent** if the forbidden set consists of only $n \times 1$ blocks.*

We define the **Horizontally Independent Fibonacci \mathbb{Z}^2 -action**, S_v , to be the system with alphabet

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

and forbidden set

$$\mathcal{F} = \left\{ \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array} \right\} .$$

■

Example 2.4 (The Domino System) Another example that will arise often is the **Domino \mathbb{Z}^2 -action** which has alphabet

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} , \boxed{2} , \boxed{3} \}$$

and forbidden set

$$\mathcal{F} = \left\{ \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline \end{array} , \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array} , \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline \end{array} , \begin{array}{|c|} \hline 1 \\ \hline 3 \\ \hline \end{array} , \begin{array}{|c|} \hline 2 \\ \hline 0 \\ \hline \end{array} , \begin{array}{|c|} \hline 3 \\ \hline 0 \\ \hline \end{array} , \right. \\ \left. \begin{array}{|c|c|} \hline 0 & 3 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 1 & 3 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 2 & 0 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 2 & 1 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 2 & 2 \\ \hline \end{array} , \begin{array}{|c|c|} \hline 3 & 3 \\ \hline \end{array} \right\} .$$

Note that the forbidden set forces the open sides of the alphabet letters to match up. This is like filling space by placing dominos on a table, thus the name.

■

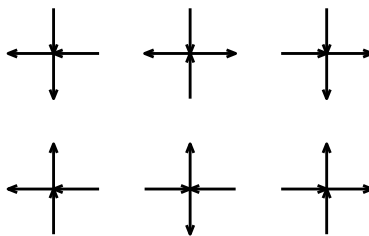


Figure 2: The six possible vertex configuration for the Square Ice Model. This is the alphabet set for this model.

Example 2.5 (Square Ice) *The **Square Ice Model** (also known as the **Six Vertex Model**) is a \mathbb{Z}^2 -action defined by assigning a configuration of arrows to each vertex of the \mathbb{Z}^2 lattice. For each point of the lattice, there are four “edges” that connect it to its four neighboring vertices. We convert these edges into arrows so that at each vertex, exactly two arrows point away from that vertex, and exactly two arrows point toward that vertex. The six possible vertex-arrow configurations can be seen in figure 2. Figure 3 shows a sample configuration for the 6-Vertex Model. A full treatment of the 6-vertex model can be found in [14] and [13].*

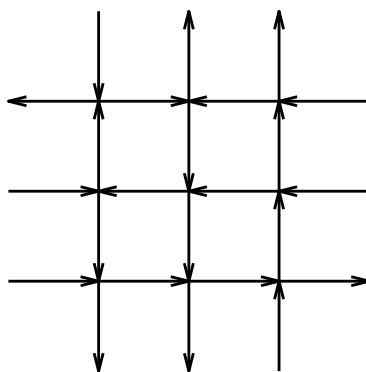


Figure 3: A sample configuration for the Square Ice Model.



Example 2.6 (The 3-coloring Model) *The square ice model is known to be equivalent to the 3-coloring \mathbb{Z}^2 -action, that is to say that there is a one to one map between the two systems. The 3-coloring model has an alphabet size of 3, with the condition that no tile can share an edge with a tile of the same name. Thus, we see that the 3-coloring system has as its alphabet and forbidden set*

$$\mathcal{A} = \{ \boxed{0}, \boxed{1}, \boxed{2} \}$$

$$\mathcal{F} = \left\{ \begin{array}{ccc} \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline \end{array}, & \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}, & \begin{array}{|c|} \hline 2 \\ \hline 2 \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array}, & \begin{array}{|c|c|} \hline 2 & 2 \\ \hline \end{array} \end{array} \right\} .$$



3 Generating the transition matrices for \mathbb{Z}^2 -actions

Described in this chapter is a substitution algorithm which generates the transition (transfer) matrices for a \mathbb{Z}^2 -action of finite type with alphabet \mathcal{A} , and forbidden set \mathcal{F} .

For the purposes of this algorithm, it is assumed that the forbidden set, \mathcal{F} , can be expressed in terms of only 2×2 blocks of tiles.

If the \mathbb{Z}^2 -action is described with a set of forbidden blocks that are bigger than 2×2 , then it must be recoded to a larger alphabet which satisfies that condition. (See “*higher block shift*” in [15].)

We will look at the transition matrices for vertical n -wide strips of tiles (transferring downward) in the system. This is often referred to as the “free boundary condition”. Each transition matrix is assumed to have rows and columns which have labels taken from \mathcal{A}^n , and are ordered lexicographically. The transition matrix for an n -wide vertical strip will be named V_n .

Note that V_n will have $|\mathcal{A}|^n$ rows and columns, labeled by the ordered elements of \mathcal{A}^n . This is because we will include the α row of V_n even if α is a forbidden block. In that case, every entry of the row α will be zero.

Let $\alpha = \alpha_1\alpha_2 \dots \alpha_n$, where $\alpha_i \in \mathcal{A}$ be the label for a row of V_n and $\beta = \beta_1\beta_2 \dots \beta_n$ be the label for a column ($\alpha, \beta \in \mathcal{A}^n$), and let $V_{n(\alpha, \beta)}$ denote the entry of V_n in the α row and the β column.

Note that for any n , V_n is completely determined by listing the number of $2 \times n$ blocks of tiles:

$$\begin{array}{|c|c|c|c|} \hline \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \hline \beta_1 & \beta_2 & \dots & \beta_n \\ \hline \end{array} = \begin{array}{|c|} \hline \alpha \\ \hline \beta \\ \hline \end{array}$$

i.e. $V_{n(\alpha,\beta)} = 0$ if $\begin{array}{|c|} \hline \alpha \\ \hline \beta \\ \hline \end{array}$ is forbidden, and $V_{n(\alpha,\beta)} = 1$ if $\begin{array}{|c|} \hline \alpha \\ \hline \beta \\ \hline \end{array}$ is an allowed block.

For a general \mathbb{Z}^2 -action, \mathcal{G} , with alphabet $\mathcal{A} = \{A, B, \dots, N\}$, we see that the binary matrix, V_2 , has the form:

	AA	AB	...	AN	BA	BB	...	BN	NA	NB	...	NN	
AA	\overline{AA}				\overline{AB}				...				\overline{AN}			
AB																
:																
:																
AN	\overline{BA}				\overline{BB}				...				\overline{BN}			
BA																
BB																
:																
:	\overline{NA}				\overline{NB}				...				\overline{NN}			
BN																
:																
:																
:	\overline{NA}				\overline{NB}				...				\overline{NN}			
NA																
NB																
:																
:	\overline{NA}				\overline{NB}				...				\overline{NN}			
NN																
:																
:																

Given $X, Y \in \mathcal{A}$, let \overline{XY} denote the submatrix of V_2 which has rows X^* and columns Y^* , as above. Then \overline{XY} describes the 2×2 blocks of the form:

$$\begin{array}{|c|c|} \hline X & * \\ \hline Y & * \\ \hline \end{array}$$

The goal is to create an algorithm which will compute V_{n+1} given V_n .

For $\alpha \in \mathcal{A}^n$ and $X \in \mathcal{A}$, let $\alpha X \in \mathcal{A}^{n+1}$ denote the (horizontal) concatenation of α and X .

If $V_{n(\alpha,\beta)} = 0$, then $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ is a forbidden block in \mathcal{G} . This implies that $\begin{bmatrix} \alpha & X \\ \beta & Y \end{bmatrix}$ will also be forbidden for any $X, Y \in \mathcal{A}$.

In this case, the $|\mathcal{A}| \times |\mathcal{A}|$ submatrix $[V_{n+1(\alpha X, \beta Y)}]_{X, Y \in \mathcal{A}}$ will have only zero entries.

In the case that $V_{n(\alpha,\beta)} = 1$, then $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ is an allowed $2 \times n$ block.

We want to know how many ways we can concatenate blocks $X, Y \in \mathcal{A}$ in the manner:

$$\begin{bmatrix} \alpha & X \\ \beta & Y \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n & X \\ \beta_1 & \beta_2 & \dots & \beta_n & Y \end{bmatrix}$$

The condition that the forbidden blocks be no larger than 2×2 implies that X and Y depend only on α_n and β_n . The corresponding submatrix,

$[V_{n+1(\alpha X, \beta Y)}]_{X, Y \in \mathcal{A}}$, then need only describe blocks of the form $\begin{bmatrix} \alpha_n & X \\ \beta_n & Y \end{bmatrix}$.

Thus, the appropriate submatrix of V_{n+1} is given by:

$$[V_{n+1(\alpha X, \beta Y)}]_{X, Y \in \mathcal{A}} = \overline{\alpha_n \beta_n}$$

Where $\overline{\alpha_n \beta_n}$ is the previously defined submatrix of V_2 .

We have now created V_{n+1} from V_n and V_2 . The V_{n+1} that we have generated also has rows and columns that are lexicographically ordered. Since lexicographic order has been preserved in the new matrix, V_{n+1} , we may again use the same substitutions that we made for V_n . Because of this, we need only find the appropriate substitutions for V_2 , and iterate those substitutions n times to generate the matrix V_{n+2} .

Note that for any system, we must actually count the allowed 2×2 blocks in order to create the matrix V_2 .

Example 3.1 *We will again look at The Hard square model from example 2.1, which had alphabet $\mathcal{A} = \{0, 1\}$, and the forbidden language*

$$\mathcal{F} = \left\{ \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} , \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array} \right\}.$$

V_1 will be the transition matrix for a 1-wide vertical strip in this system.

Thus,

$$V_1 = \begin{array}{c} 0 \\ 1 \end{array} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

The rows and columns of V_1 are labeled $0, 1 \in \mathcal{A}$. Note that this is also the transition matrix for the 1-dimensional Fibonacci shift system, as expected.

An inspection of the allowed 2×2 blocks reveals that:

$$V_2 = \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & | & 1 & 0 \\ 1 & 0 & | & 1 & 0 \\ \hline 1 & 1 & | & 0 & 0 \\ 0 & 0 & | & 0 & 0 \end{bmatrix} = \begin{bmatrix} \overline{00} & | & \overline{01} \\ \hline \overline{00} & | & \overline{01} \end{bmatrix}$$

With the rows and columns being lexicographically ordered elements of \mathcal{A}^2 .

Now:

$$V_{2_{(00,00)}} = 1 \Rightarrow V_{3_{(00X,00Y)}} = \overline{00}$$

$$V_{2_{(00,01)}} = 1 \Rightarrow V_{3_{(00X,01Y)}} = \overline{01}$$

$$V_{2_{(00,10)}} = 1 \Rightarrow V_{3_{(00X,10Y)}} = \overline{00}$$

$$V_{2_{(00,11)}} = 0 \Rightarrow V_{3_{(00X,11Y)}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \overline{11}$$

At this point, we can exploit the fact that $\overline{11}$ is a zero matrix. If no such submatrix of V_2 exists, then we must create one.

Continuing with the substitutions, we get:

$$V_3 = \begin{array}{l} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{cc|cc} \overline{00} & \overline{01} & \overline{00} & \overline{11} \\ \overline{10} & \overline{11} & \overline{10} & \overline{11} \\ \hline \overline{00} & \overline{10} & \overline{11} & \overline{11} \\ \overline{11} & \overline{11} & \overline{11} & \overline{11} \end{array} \right]$$

Now note that since the matrix V_3 is lexicographically ordered, we can again make the same type of substitutions to obtain V_4 .

■

It is of special note that in example 3.1, it is not necessary now to refer back to the submatrices of V_2 .

Recall that:

$$V_2 = \begin{bmatrix} \overline{00} & \overline{01} \\ \overline{10} & \overline{11} \end{bmatrix}$$

And we have, in essence, created the substitution rule:

$$\overline{00} \longrightarrow \begin{bmatrix} \overline{00} & \overline{01} \\ \overline{01} & \overline{11} \end{bmatrix}$$

$$\overline{01} \longrightarrow \begin{bmatrix} \overline{00} & \overline{11} \\ \overline{10} & \overline{11} \end{bmatrix}$$

$$\overline{10} \longrightarrow \begin{bmatrix} \overline{00} & \overline{01} \\ \overline{11} & \overline{11} \end{bmatrix}$$

$$\overline{11} \longrightarrow \begin{bmatrix} \overline{11} & \overline{11} \\ \overline{11} & \overline{11} \end{bmatrix}$$

Starting with the matrix V_2 , and iterating this substitution n times, we obtain V_{n+2} , the transition matrix for an $(n+2)$ -wide vertical strip (moving downward).

Once V_n is known, the sum of all nonzero entries in the matrix V_n^{n-1} is exactly the number of allowed $n \times n$ blocks for the system \mathcal{G}

Since the matrix V_{n+1} generated by this substitution still has lexicographical ordering of rows and columns, and is the same substitution at every level, we can improve our notation by partitioning the matrix V_n by writing

$$V_n = \left(\begin{array}{c|c} \overline{00}_n & \overline{01}_n \\ \hline \overline{10}_n & \overline{11}_n \end{array} \right).$$

And defining our substitutions as

$$\overline{00}_{n+1} = \begin{pmatrix} \overline{00}_n & \overline{01}_n \\ \overline{10}_n & 0 \end{pmatrix} \quad ; \quad \overline{01}_{n+1} = \begin{pmatrix} \overline{00}_n & 0 \\ \overline{10}_n & 0 \end{pmatrix} \quad ;$$

$$\overline{10}_{n+1} = \begin{pmatrix} \overline{00}_n & \overline{01}_n \\ 0 & 0 \end{pmatrix}.$$

We will find that this new notation will be convenient for later calculations.

However throughout this paper, we will continue to use both notations, using that which best suits our needs at the time.

We also note that after we substitute a few times, we expect to have a lot of zero entries in the matrix V_n . In particular, there will be many rows and columns which contain only zero entries.

We can, at any time, eliminate rows and columns of zeros, without compromise to the spectral radius or any relevant calculations, but at the cost of homogeneity in the substitution process. This method was used in [17] for the Hard Square system.

To do this, we define a new matrix, \check{V}_n which is the matrix V_n stripped of any zero rows or columns.

Thus, for the Hard Square system,

$$\check{V}_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

and,

$$\check{V}_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

At this point, keeping the zero rows and columns is a matter of having an easily-defined substitution. We could define a (more complicated) substitution algorithm which eliminates zero rows and columns altogether, however, as we will see later, keeping the zero rows and columns will allow us to perform a few computations that we could not do otherwise.

4 A reflection operator for \mathbb{Z}^2 -actions

We define a \mathbb{Z}^2 -action, \mathcal{G} with alphabet \mathcal{A} and finite forbidden set \mathcal{F} to be *empty action* if \mathcal{G} is empty. Otherwise, we say that \mathcal{G} is a *valid* \mathbb{Z}^2 -action.

Until now, we have only considered the binary matrices, V_n , which describe n -wide vertical strips (traveling downward) for \mathbb{Z}^2 -actions. There is no reason that we could not use n -tall infinite (right-traveling) horizontal strips.

At this point, we will introduce the notation H_n for the binary transfer matrix which describes the n -tall horizontal strips, moving to the right. In other words, for a given \mathbb{Z}^2 -action, \mathcal{G} , if $H_{n(\alpha,\beta)} = 1$, then the $n \times 2$ block:

$$\boxed{\alpha \mid \beta} = \begin{array}{|c|c|} \hline \alpha_1 & \beta_1 \\ \hline \alpha_2 & \beta_2 \\ \hline \vdots & \vdots \\ \hline \alpha_n & \beta_n \\ \hline \end{array}$$

is an allowed block in \mathcal{G} . If this block is forbidden then it must be the case that $H_{n(\alpha,\beta)} = 0$.

Let \mathcal{G} be a non-empty \mathbb{Z}^2 -action, with alphabet \mathcal{A} . Let $n = |\mathcal{A}|$. Let $\mathbf{B}(m)$ denote the set of $m \times m$ binary matrices.

Definition 4.1 Let $A = [a_{i,j}] \in \mathbf{B}(n^2)$. For $1 \leq k \leq n^2$, define functions

$\Phi_k : \mathbf{B}(n^2) \longrightarrow \mathbf{B}(n)$ by

$$\Phi_k(A) = \begin{bmatrix} a_{k,1} & a_{k,2} & \dots & a_{k,n} \\ a_{k,n+1} & a_{k,n+2} & \dots & a_{k,2n} \\ & & \ddots & \\ a_{k,n^2-n+1} & & \dots & a_{k,n^2} \end{bmatrix}.$$

Then each map Φ_k is a re-arrangement of the k^{th} row of A into a square matrix. (Cut the k^{th} row into n vectors, then stack those vectors to make a $n \times n$ matrix.)

Definition 4.2 Define the **reflection operator**, $\Psi : \mathbf{B}(n^2) \longrightarrow \mathbf{B}(n^2)$ by

$$\Psi(A) = \begin{bmatrix} \Phi_1(A) & \Phi_2(A) & \dots & \Phi_n(A) \\ \Phi_{n+1}(A) & \Phi_{n+2}(A) & \dots & \Phi_{2n}(A) \\ & & \ddots & \\ \Phi_{n^2-n+1} & & \dots & \Phi_{n^2}(A) \end{bmatrix}.$$

We then have that:

Theorem 4.1 Let \mathcal{G} be a \mathbb{Z}^2 -action with alphabet \mathcal{A} , $\Psi : \mathbf{B}(n^2) \longrightarrow \mathbf{B}(n^2)$ be the reflection operator defined above, and let $n = |\mathcal{A}|$. If V_2 and H_2 are the vertical and horizontal transfer matrices of \mathcal{G} which describe the 2-wide and 2-tall infinite strips, then:

- (1) Ψ is 1:1 and onto.
- (2) $\Psi = \Psi^{-1}$.
- (3) $\Psi(V_2) = H_2$.

(Note that (2) and (3) are the reason that we refer to this as the **reflection operator**)

Proof: Examination of the nature of Ψ reveals that the row $V_{2_{(AB, **)}}$ is mapped to the submatrix $\overline{AB} = [\Psi(V_2)]_{(A^*, B^*)}$ as follows:

For an ordered alphabet, $\mathcal{A} = \{A, B, \dots, N\}$, we will denote by $\varrho(X)$ the (integer) position of X in \mathcal{A} . For example, if \mathcal{A} is the ordered set $\{H, F, W, Q, D\}$, then $\varrho(Q) = 4$.

Let $a = \varrho(A)$, $b = \varrho(B)$, $c = \varrho(C)$, and $d = \varrho(D)$. The element $V_{2_{(AB, CD)}}$ is then in the $a(n-1) + b$ row and $c(n-1) + d$ column of V_2 .

If $x = a(n-1) + b$ and $y = c(n-1) + d$, we see that $[\Phi_x(V_2)]_{(i, d)} = [A]_{(x, y)}$ for $1 \leq i \leq n$.

Also, the $\Phi_x(V_2)$ are ordered inside of $\Psi(V_2)$ such that

$$[\Psi(V_2)]_{(a(n-1)+c, b(n-1)+d)} = [V_2]_{(x, y)}.$$

This is then expressed as the equation

$$[V_2]_{(AB, CD)} = [\Psi(V_2)]_{(AC, BD)}.$$

(1) and (2) immediately follow.

For (3), we note that $V_{2_{(AB, CD)}} = H_{2_{(AC, BD)}}$ since both determine whether the 2×2 block,

A	B
C	D

is allowed or not.

The proof is then completed. ■

We will now give an illustration of the proof.

Let \mathcal{G} be a \mathbb{Z}^2 -action with alphabet $\mathcal{A} = \{A, B\}$. Then V_2 has the form:

$$V_2 = \begin{array}{c|cccc} & AA & AB & BA & BB \\ \hline AA & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ \hline AB & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \hline BA & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ \hline BB & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{array}$$

Then:

$$\Psi(V_2) = H_2 = \begin{array}{c|cccc} & AA & AB & BA & BB \\ \hline AA & a_{1,1} & a_{1,2} & a_{2,1} & a_{2,2} \\ \hline AB & a_{1,3} & a_{1,4} & a_{2,3} & a_{2,4} \\ \hline BA & a_{3,1} & a_{3,2} & a_{4,1} & a_{4,2} \\ \hline BB & a_{3,3} & a_{3,4} & a_{4,3} & a_{4,4} \end{array}$$

We see that the row AB of V_2 is transformed into the submatrix \overline{AB} of

$$\Psi(V_2) = H_2 .$$

Theorem 4.2 *If V_2 describes a non-empty shift system, then the following are equivalent:*

- (1) *The row XY of V_2 has only zero entries.*
- (2) *The column XY of V_2 has only zero entries.*

Proof: Let V_2 describe a valid shift system. Let XY be a row of V_2 which has only zero entries. Then there is no vertical transition from XY to AB for any $AB \in \mathcal{A}^2$. In other words, there cannot be a block of the form:

X	Y
A	B

Now suppose that the column XY has a nonzero entry. This would imply that there exist $CD \in \mathcal{A}^2$ such that $V_{2(CD,XY)} = 1$. That is to say that the block:

C	D
X	Y

is allowed. But such a block forbids any further vertical transition. Thus, V_2 does not describe a valid \mathbb{Z}^2 -action. The reverse implication is similar.

■

Corollary 4.1 *If H_2 describes a non-empty shift system, then the following are equivalent:*

- (1) *The row XY of H_2 has only zero entries.*
- (2) *The column XY of H_2 has only zero entries.*

The proof is the same as that of the theorem, interchanging up and down for left and right transitions.

Corollary 4.2 *V_2 describes a non-empty \mathbb{Z}^2 -action $\Leftrightarrow \Psi(V_2)$ describes a non-empty \mathbb{Z}^2 -action.*

Corollary 4.3 *If V_2 describes a non-empty \mathbb{Z}^2 -action and $\Psi(V_2) = H_2$, then the following are equivalent:*

- (1) *The row XY of H_2 is all zeros.*
- (2) *The column XY of H_2 is all zeros.*
- (3) *The submatrix \overline{XY} of V_2 is all zeros.*
- (4) $V_{2_{(AX, BY)}} = 0 \quad \forall A, B \in \mathcal{A}$

Proof: This is a direct result of the previous corollary combined with the properties of the map Ψ .

■

5 Computing entropy for \mathbb{Z}^2 -actions

If κ is the entropy for a \mathbb{Z}^2 -action, \mathcal{G} , then we expect the number of $n \times n$ square blocks to grow asymptotically according to $|\mathcal{B}_{n,n}(\mathcal{G})| \sim 2^{\kappa n^2}$ in the sense that $0 \leq \lim_{n \rightarrow \infty} \frac{1}{n^2} \log |\mathcal{B}_{n,n}(\mathcal{G})| = \kappa < \infty$.

However, in the previously described construction, each V_n was the transition matrix for a 1-dimensional shift system. If h_n is the entropy for the 1-dimensional vertical strip described by V_n , then as m becomes large, $|\mathcal{B}_{m,n}(\mathcal{G})| \sim 2^{h_n m}$.

Since we know that $h_n \sim \kappa n$, it is natural to ask under what circumstances is it true that $h_{n+1} - h_n \sim \kappa$. We know that h_n is given by $\log(\lambda_{V_n})$, where λ_{V_n} is the Perron eigenvalue of V_n . Thus we are asking when it is true that

$$\lim_{n \rightarrow \infty} \log\left(\frac{\lambda_{V_{n+1}}}{\lambda_{V_n}}\right) = \kappa.$$

The seeming convergence of the sequence of ratios of eigenvalues was first discovered by the author in 2000. In a private conversation, Boris Solomyak at the University of Washington relayed that a former student of his, D. Petry, had taken note of this sequence in the late 1980's. In 1994, Dr. Solomyak included this problem in a grant proposal to the National Science Foundation.

Further conversations with Robert Burton (Oregon State University), Paul Shields (University of Toledo), Chris Hoffman (University of Washington),

Doug Lind (University of Washington), and others have led the author to believe that this question has existed in the folklore of the subject as early as the late 1970's or early 1980's. (Private conversations: [20] , [19], [16], [11].)

In practice, we see that $\frac{\lambda_{V_{n+1}}}{\lambda_{V_n}}$ converges much faster than counting the $\mathcal{B}_{n,n}(\mathcal{G})$ blocks found in the entries of the matrix V_n^{n-1} . In this section, we give a proof that $\frac{\lambda_{n+1}}{n}$ converges to the correct entropy for a certain class of \mathbb{Z}^d -actions. The result is then stated in Theorem 5.4.

If the sequence $\frac{\lambda_{V_{n+1}}}{\lambda_{V_n}}$ does indeed converge, then it is trivial that it converges to the correct entropy. This is due to the following theorem from [18].

Theorem 5.1 *For any sequence of complex numbers, a_n ,*

$$\liminf \left| \frac{a_{n+1}}{a_n} \right| \leq \liminf |a_n|^{1/n} \leq \limsup |a_n|^{1/n} \leq \limsup \left| \frac{a_{n+1}}{a_n} \right|.$$

Thus, if we get convergence for the ratios, it is then true that

$$\lim_{n \rightarrow \infty} \log \frac{\lambda_{n+1}}{\lambda_n} = \lim_{n \rightarrow \infty} \log \lambda_n^{1/n} = \kappa$$

where κ is the correct entropy.

To see why the convergence of ratios of successive λ_n is of such interest, here is an example:

The `genalg.m` MATLAB program (see appendix A) was used to compute the V_n matrices for the Hard Square system. Then $|\mathcal{B}_{n,n}|$ was computed by taking

V_n to the $n - 1$ power and summing over the entries. The number of allowable n -squares was then computed as $|\mathcal{B}_{n,n}|^{1/n}$. It was found that

$$|\mathcal{B}_{11,11}|^{1/11} = 1.52229012974517.$$

In a different computation, the V_n were generated using the **genalg.m** program as before. However, the Perron eigenvalue was found for V_n by issuing an **eigs** command in MATLAB and finding the maximum eigenvalue for each V_n . It was found that

$$\frac{\lambda_9}{\lambda_8} = 1.50304808228931.$$

The correct entropy as computed by Baxter in [1] to forty-four decimal places is 1.5030480824753322643220663294755536893857810.

We see that the first approximation to the entropy is only good to two decimal places, but the ratio of eigenvalues algorithm gives ten decimal places of accuracy even while using fewer iterations. As extra incentive, the first computation took nearly twenty minutes to get such a poor approximation, while the second ran in less than one minute on the same computer!

Comparison of these two algorithms used on various other \mathbb{Z}^2 -actions (Square Ice, Dominos, etc.) give similar results. Truly the ratio of eigenvalues is the clear choice for the numerical calculation of entropy (for now).

However, as mentioned previously, there are \mathbb{Z}^2 -actions for which the ratios of eigenvalues do not converge. The following is an example of such a system.

Example 5.1 Consider the \mathbb{Z}^2 -action, \mathcal{G} , with alphabet $\mathcal{A} = \{0, 1, 2\}$ and the following construction:

- (1) Every other row (i.e. even or odd rows) is a sequence in $\{0, 1\}^{\mathbb{Z}}$.
- (2) The rows in between those described in (1) are $\{2\}^{\mathbb{Z}}$.

This system can be described by the forbidden block set:

$$\mathcal{F} = \left\{ \begin{bmatrix} 0 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 2 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}$$

This system is better understood by looking at an actual block. A typical block may look like:

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

Let V_n denote the (down-traveling) vertical transfer matrix for an n -wide strip in \mathcal{G} , and H_n denote the (right-traveling) horizontal transfer matrix for an n -tall strip. We see that:

$$V_2 = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \right]$$

A simple count of $(m \times n)$ -blocks shows that if m (the number of horizontal rows) is even, then:

$$|\mathcal{B}_{m,n}(\mathcal{G})| = 2^{\frac{mn}{2} + 1}$$

And if m is odd:

$$|\mathcal{B}_{m,n}(\mathcal{G})| = 2^{\frac{(m+1)n}{2}} + 2^{\frac{(m-1)n}{2}}$$

Thus, we have for the entropy of \mathcal{G} :

$$h(\mathcal{G}) = \lim_{m,n \rightarrow \infty} \frac{1}{mn} \log |\mathcal{B}_{m,n}(\mathcal{G})| = \frac{1}{2}$$

Now, if we call h_{V_n} the entropy for an n -wide vertical strips, we get:

$$h_{V_n}(\mathcal{G}) = \lim_{m \rightarrow \infty} \frac{1}{m} \log |\mathcal{B}_{m,n}(\mathcal{G})| = \frac{n}{2}$$

Thus, the Perron eigenvalue for V_n must be $\lambda_{V_n} = 2^{\frac{n}{2}}$.

This gives us that

$$\lim_{n \rightarrow \infty} \log \left(\frac{\lambda_{V_{n+1}}}{\lambda_{V_n}} \right) = \lim_{n \rightarrow \infty} \log \left(\frac{2^{\frac{n+1}{2}}}{2^{\frac{n}{2}}} \right) = \frac{1}{2}$$

We have then in this case that the ratio of eigenvalues is converging to the correct entropy.

Now if we let $h_{H_m}(\mathcal{G})$ denote the entropy for an m -tall horizontal strip, for even m we get:

$$h_{H_m}(\mathcal{G}) = \lim_{n \rightarrow \infty} \frac{1}{n} \log(2^{\frac{mn}{2}+1}) = \frac{m}{2}$$

And for m odd we get:

$$h_{H_m}(\mathcal{G}) = \lim_{n \rightarrow \infty} \frac{1}{n} \log\left(2^{\frac{(m+1)n}{2}} + 2^{\frac{(m-1)n}{2}}\right) = \frac{m+1}{2}$$

This gives us that for even m , $\lambda_{H_m} = 2^{\frac{m}{2}}$, and for m odd, $\lambda_{H_m} = 2^{\frac{m+1}{2}}$.

We then see that for even m ,

$$\frac{\lambda_{H_{m+1}}}{\lambda_{H_m}} = \frac{2^{\frac{m+2}{2}}}{2^{\frac{m}{2}}} = 2$$

And for m odd:

$$\frac{\lambda_{H_{m+1}}}{\lambda_{H_m}} = \frac{2^{\frac{m+1}{2}}}{2^{\frac{m+1}{2}}} = 1$$

This yields the sequence:

$$\left[\frac{\lambda_{H_{m+1}}}{\lambda_{H_m}} \right]_{m \in \mathbb{N}} = 1, 2, 1, 2, 1, 2, \dots$$

$$\implies \left[\log\left(\frac{\lambda_{H_{m+1}}}{\lambda_{H_m}}\right) \right]_{m \in \mathbb{N}} = 0, 1, 0, 1, \dots$$

This shows us that the ratio of Perron eigenvalues does not converge. Also, neither the \limsup_m nor the \liminf_m of the sequence yields the correct entropy.

■

Remark: We note that none of the H_n matrices are irreducible, and all of the V_n are periodic. By the construction of this system, we see that we are, in fact, making many “discrete jumps” in the entropy of the horizontal strips. That is to say that by increasing the height of an infinite horizontal strip from n to $n + 1$, the entropy increases only when n is even. If we start with an odd height, and increase to the next even height, the entropy remains the same.

It is notable, however, that the average of the \limsup and the \liminf of this sequence is the correct entropy. This leads us to believe that the *Cesaro limit* of such a sequence will give the correct entropy of the system.

Theorem 5.2 *If \mathcal{G} is a \mathbb{Z}^2 -action, and $h_n = \log(\lambda_n)$ is the entropy of an n -wide (vertical) restriction of \mathcal{G} , then:*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \log \left[\frac{\lambda_i}{\lambda_{i-1}} \right] = h$$

Where h is the entropy of the system, \mathcal{G}

Proof:

$$\begin{aligned}
& \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \log \left[\frac{\lambda_i}{\lambda_{i-1}} \right] \\
&= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n [\log(\lambda_i) - \log(\lambda_{i-1})] \\
&= \lim_{n \rightarrow \infty} \frac{1}{n} [\log(\lambda_n) - \log(\lambda_1)] \\
&= \lim_{n \rightarrow \infty} \frac{1}{n} \log \lambda_n = h
\end{aligned}$$

■

Note that for the Hard Square system given earlier, each of the matrices V_n was irreducible and aperiodic over its non-zero rows and columns. That is to say that if \check{V}_n is the matrix obtained by deleting from V_n any row or column which consists of entirely zeros, then \check{V}_n is irreducible and aperiodic. The presence of the “free square”, $\boxed{0}$, guarantees that the system is both irreducible and aperiodic.

These are precisely the conditions that we need in order to prove that the ratio of eigenvalues method will yield the correct entropy.

We start with a statement of a theorem from [15] (Page 130) (with a convenient adaptation of indices and terminology):

Theorem 5.3 *Let A be a non-negative (and non-zero), irreducible, aperiodic matrix with Perron eigenvalue λ . Let \vec{v} and \vec{w} be the right and left Perron eigenvectors for A : $A\vec{v} = \lambda\vec{v}$ and $\vec{w}A = \lambda\vec{w}$ and normalized so that the dot product $\vec{w} \cdot \vec{v} = 1$. Then for each i, j ,*

$$[A^n]_{i,j} = [(\vec{v}_i \cdot \vec{w}_j) + \rho_n(i, j)] \lambda^n$$

where $\rho_n(i, j) \rightarrow 0$ as $n \rightarrow \infty \forall i, j$, and $\vec{v}_i \cdot \vec{w}_j$ denotes the regular multiplication of the i^{th} index of \vec{v} with the j^{th} index of \vec{w} .

Given a \mathbb{Z}^2 -action (Hard Squares, Square Ice, etc.), which has irreducible, aperiodic transfer matrices, we let V_n denote the vertical transfer matrix for an n -wide strip, and H_k will denote the horizontal transfer matrix for a k -tall strip.

As usual, we will let λ_n be the Perron eigenvalue of V_n . As in the terminology of the above theorem, we will let \vec{v}_n^V and \vec{w}_n^V denote (respectively) the right and left Perron eigenvectors of V_n and \vec{v}_n^H , \vec{w}_n^H denote (respectively) the right and left Perron eigenvectors of H_n .

For the following, we will only consider 2-D systems which have irreducible and aperiodic V_n and H_n for each n .

We now write the above theorem in the current context,

$$[V_n^k]_{i,j} = [(\vec{w}_{n,i}^V \cdot \vec{v}_{n,j}^V) + \rho_{n,k}^V(i, j)] \lambda_n^k$$

where $\rho_{n,k}^V(i, j) \rightarrow 0$ as $k \rightarrow \infty$.

Now,

$$\lambda_n = \lim_{k \rightarrow \infty} |\mathcal{B}_{k,n}|^{\frac{1}{k}}$$

where $\mathcal{B}_{k,n}$ denotes the set of allowed $k \times n$ blocks in our system. We then have

$$\begin{aligned} \lambda_n &= \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} [H_k^{n-1}]_{i,j} \right\}^{\frac{1}{k}} \\ &= \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} [(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H) + \rho_{k,n-1}^H(i, j)] \lambda_k^{n-1} \right\}^{\frac{1}{k}} \\ &= \lim_{k \rightarrow \infty} \left\{ \lambda_k^{\frac{1}{k}} \right\}^{n-1} \cdot \left\{ \sum_{i,j} [(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H) + \rho_{k,n-1}^H(i, j)] \right\}^{\frac{1}{k}}. \end{aligned}$$

Both terms in the curly braces above converge, and thus we have that for every n ,

$$\lambda_n = \mathcal{H}^{n-1} \cdot \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} [(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H) + \rho_{k,n-1}^H(i, j)] \right\}^{\frac{1}{k}} \quad (2)$$

where $\mathcal{H} = 2^h$ and h is the entropy for the system in question.

($\mathcal{H} = \lim_{n \rightarrow \infty} (\lambda_n)^{\frac{1}{n}}$.) We then divide \mathcal{H}^{n-1} to get

$$\frac{\lambda_n}{\mathcal{H}^{n-1}} = \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} [(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H) + \rho_{k,n-1}^H(i, j)] \right\}^{\frac{1}{k}}. \quad (3)$$

We know that the sequence $\lambda_n^{\frac{1}{n}}$ is a monotonically decreasing sequence that converges to \mathcal{H} (see [15] and [10]). Thus, we will use the convention that $\lambda_n^{\frac{1}{n}} = \mathcal{H} + \varepsilon_n$ where $\varepsilon_n \downarrow 0$ as $n \rightarrow \infty$. The left hand side of the above becomes

$$\frac{\lambda_n}{\mathcal{H}^{n-1}} = \mathcal{H} \cdot \frac{\lambda_n}{\mathcal{H}^n} = \mathcal{H} \cdot \left(\frac{\lambda_n^{\frac{1}{n}}}{\mathcal{H}} \right)^n = \mathcal{H} \cdot \left(\frac{\mathcal{H} + \varepsilon_n}{\mathcal{H}} \right)^n = \mathcal{H} \cdot \left(1 + \frac{\varepsilon_n}{\mathcal{H}} \right)^n$$

which is true for every n . Taking limits with n ,

$$\lim_{n \rightarrow \infty} \frac{\lambda_n}{\mathcal{H}^{n-1}} = \mathcal{H} \cdot \lim_{n \rightarrow \infty} \left(1 + \frac{\varepsilon_n}{\mathcal{H}} \right)^n$$

Since $\varepsilon_n \downarrow 0$, the right hand side above converges with n . Therefore, the left hand side also converges with n . Using this in equation (3), we get that

$$\lim_{n \rightarrow \infty} \frac{\lambda_n}{\mathcal{H}^{n-1}} = \lim_{n \rightarrow \infty} \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} [(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H) + \rho_{k,n-1}^H(i,j)] \right\}^{\frac{1}{k}}. \quad (4)$$

Of particular importance is the fact that the right hand side of the above (4) converges with n .

From equation (2), we have that

$$\begin{aligned}
\frac{\lambda_n}{\lambda_{n-1}} &= \frac{\mathcal{H}^{n-1} \cdot \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-1}^H(i,j) \right] \right\}^{\frac{1}{k}}}{\mathcal{H}^{n-2} \cdot \lim_{k \rightarrow \infty} \left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-2}^H(i,j) \right] \right\}^{\frac{1}{k}}} \\
&= \mathcal{H} \cdot \lim_{k \rightarrow \infty} \frac{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-1}^H(i,j) \right] \right\}^{\frac{1}{k}}}{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-2}^H(i,j) \right] \right\}^{\frac{1}{k}}}.
\end{aligned}$$

(Note that we now have a convergence bound on the error for $\frac{\lambda_n}{\lambda_{n-1}}$)

Now, taking limits with n , we get

$$\begin{aligned}
\lim_{n \rightarrow \infty} \left(\frac{\lambda_n}{\lambda_{n-1}} \right) &= \lim_{n \rightarrow \infty} \left(\mathcal{H} \cdot \lim_{k \rightarrow \infty} \frac{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-1}^H(i,j) \right] \right\}^{\frac{1}{k}}}{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-2}^H(i,j) \right] \right\}^{\frac{1}{k}}} \right) \\
&= \mathcal{H} \cdot \lim_{n \rightarrow \infty} \lim_{k \rightarrow \infty} \left(\frac{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-1}^H(i,j) \right] \right\}^{\frac{1}{k}}}{\left\{ \sum_{i,j} \left[\left(\vec{\mathbf{w}}_{k,i}^H \cdot \vec{\mathbf{v}}_{k,j}^H \right) + \rho_{k,n-2}^H(i,j) \right] \right\}^{\frac{1}{k}}} \right) \quad (5)
\end{aligned}$$

From equation (4), we know that the numerator and denominator on the right hand side of equation (5) converge to a number with n , and thus we have that

$$\lim_{n \rightarrow \infty} \left(\frac{\lambda_n}{\lambda_{n-1}} \right) = \mathcal{H} \cdot 1 = \mathcal{H}.$$

We have just proved the following theorem.

Theorem 5.4 *If \mathcal{G} is a \mathbb{Z}^d -action with the property that its restricted transfer matrices, \check{V}_n and \check{H}_n are irreducible and aperiodic for each n , and if λ_n denotes the Perron eigenvalue of the vertical restriction, V_n , then*

$$\lim_{n \rightarrow \infty} \ln \left(\frac{\lambda_n}{\lambda_{n-1}} \right) = h$$

where h is the entropy of \mathcal{G} .

Corollary 5.1 *For the Hard Square \mathbb{Z}^2 -action, the limit of the sequence of ratios of Perron eigenvalues is equal to the entropy, $\lim_{n \rightarrow \infty} \frac{\lambda_{n+1}}{\lambda_n} = h$.*

Proof:

The existence of the free square, \square_0 , guarantees that each V_n for the Hard Square system is both irreducible and aperiodic, and thus our theorem applies to the Hard Square system.

■

6 Asymptotic Analysis of the Substitution Algorithm and its Relationship to Shifts of Finite Type

In this section, we examine the restricted vertical transfer matrices, V_n , for a \mathbb{Z}^2 -action (as defined in Chapter 3), and their Perron eigenvectors, in order to better understand the asymptotic behavior of the substitution algorithm.

For illustrative purposes, we will regularly refer to two \mathbb{Z}^2 -actions which are closely related to the Hard Square \mathbb{Z}^2 -action and the one-dimensional Fibonacci shift of finite type.

We will define \mathcal{S}_h to be the \mathbb{Z}^2 -action which consists of stacks of horizontal one-dimensional Fibonacci shifts with no vertical correlations, a property which we will refer to as *vertical independence*. More precisely, a *vertically independent system* is one in which the forbidden set consists of only $1 \times n$ blocks.

Define \mathcal{S}_h to be the \mathbb{Z}^2 -action with alphabet:

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

And forbidden set:

$$\mathcal{F} = \{ \boxed{11} \}$$

The corresponding transfer matrix for this system is given by:

$$V_2 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array} \right]$$

We will also make use of the system which consists of infinite vertical strips of Fibonacci shifts, with no horizontal correlations. We define a \mathbb{Z}^2 -action to be *horizontally independent* if the forbidden set consists of only $n \times 1$ blocks.

We define the \mathbb{Z}^2 -action \mathcal{S}_v to be the system with alphabet:

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

And forbidden set:

$$\mathcal{F} = \left\{ \begin{array}{c} \boxed{1} \\ \boxed{1} \end{array} \right\}$$

The corresponding matrix for this system is given by:

$$V_2 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & 0 \end{array} \right]$$

We now start our analysis with an examination of the property that the substitution algorithm not only generates the transition matrices for k -wide vertical strips of a \mathbb{Z}^2 -action, but it also keeps track of the actual allowed k -words in a one-dimensional shift of finite type.

We illustrate this property with the following example, which describes the 2-tall horizontal restriction of the Hard Square system:

Let \mathcal{G} be the (one-dimensional) shift of finite type with alphabet:

$$\mathcal{A} = \left\{ \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline \end{array} ; \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array} ; \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline \end{array} \right\}$$

and forbidden set:

$$\mathcal{F} = \left\{ \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array} ; \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 0 & 0 \\ \hline \end{array} \right\}$$

A typical point in \mathcal{G} might look like:

...	0	0	1	0	1	...
...	1	0	0	1	0	...

Note now that \mathcal{G} is identical to the 2-tall horizontal restriction of the Hard Square \mathbb{Z}^2 -action.

The transition matrix for \mathcal{G} is:

$$T_{\mathcal{G}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

In order to make the substitution algorithm work, we need the "full" alphabet.

Thus, we add the *forbidden letter*, $\begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}$ to the end of the alphabet.

Thus, our new transfer matrix is:

$$T_{\mathcal{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Which we recognize as V_2 for the Hard Square system.

If we define the usual substitutions for the hard square, we get:

$$V_3 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right]$$

Note that the $[3, 2]$ entry of V_3 (which is equivalent to the $[010, 001]$ entry for V_3 in the 2-dimensional setting) is one, and thus the 3-word:

...	0	1	0	...
...	0	0	1	...

Is an allowable word for the 1-dimensional shift \mathcal{G} .

We also see that the $[2, 6]$ entry of V_3 (which is equivalent to the $[001, 101]$ entry for V_3 in the 2-dimensional setting) is zero, and thus the 3-word:

...	0	0	1	...
...	1	0	1	...

Is not an allowed word for \mathcal{G} .

Continuing the substitution, we see that every non-zero entry in V_k corresponds to an allowed k -word in the one-dimensional shift, \mathcal{G} , and that every zero entry corresponds to a forbidden word of length k .

Thus we can think of the substitution algorithm as generating transition matrices for a \mathbb{Z}^2 -action, but it can also be thought of as keeping track of the names of allowed blocks in a one-dimensional shift of finite type.

From this, we can see that for any \mathbb{Z}^2 -action and any $k \geq 1$:

$$\sum_{\alpha, \beta \in \mathcal{A}^{k-1}} (V_k)_{[\alpha, \beta]} = \sum_{i=1}^{|\mathcal{A}|} (H_2^{k-1})_{[XY, i]} \quad (6)$$

Another important observation is that the substitution algorithm will yield the correct entropy regardless of whether the initial "seed matrix", V_2 , is binary.

It only matters where the zero entries are placed and that the initial matrix is non-negative.

Let M be an $n^2 \times n^2$ (not necessarily binary) matrix with non-negative entries.

Define an $n^2 \times n^2$ binary matrix, called the *indicator matrix of M* , $\mathbb{I}(M)$ by:

$$\mathbb{I}(M)_{[A, B]} = \begin{cases} 0 & M_{[A, B]} = 0 \\ 1 & M_{[A, B]} > 0 \end{cases}$$

Lemma 6.1 *Let M be a non-negative matrix and let $V_2 = \mathbb{I}(M)$. Suppose that V_2 defines a valid \mathbb{Z}^2 -action, then we can define substitutions on V_2 according to chapter 3.*

If we apply the same substitutions to M , and call M_k the matrix obtained by applying this substitution $k - 2$ times to M , then:

$$\lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho(M_k)] = \lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho(V_k)] = h$$

Where $\rho(V_k)$ denotes the spectral radius of the matrix V_k , and h is the entropy of the \mathbb{Z}^2 -action defined by V_2 .

Proof: Define $\alpha = \max\{M_{[A,B]} > 0\}$ and $\beta = \min\{M_{[A,B]} > 0\}$. Then:

$$\begin{aligned} \beta \rho(V_k) &\leq \rho(M_k) \leq \alpha \rho(V_k) \\ \implies [\beta \rho(V_k)]^{1/k} &\leq [\rho(M_k)]^{1/k} \leq [\alpha \rho(V_k)]^{1/k} \\ \implies [\beta]^{1/k} [\rho(V_k)]^{1/k} &\leq [\rho(M_k)]^{1/k} \leq [\alpha]^{1/k} [\rho(V_k)]^{1/k} \\ \implies \lim_{k \rightarrow \infty} [\beta]^{1/k} [\rho(V_k)]^{1/k} &\leq \lim_{k \rightarrow \infty} [\rho(M_k)]^{1/k} \leq \lim_{k \rightarrow \infty} [\alpha]^{1/k} [\rho(V_k)]^{1/k} \\ \implies \lim_{k \rightarrow \infty} [\rho(V_k)]^{1/k} &\leq \lim_{k \rightarrow \infty} [\rho(M_k)]^{1/k} \leq \lim_{k \rightarrow \infty} [\rho(V_k)]^{1/k} \\ \implies \lim_{k \rightarrow \infty} [\rho(M_k)]^{1/k} &= \lim_{k \rightarrow \infty} [\rho(V_k)]^{1/k} \\ \lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho(M_k)] &= \lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho(V_k)] \end{aligned}$$

■

This lemma is better illustrated with an example. For the case of the Hard Square system:

$$V_2 = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) = \left(\begin{array}{c|c} \mathbf{A}_2 & \mathbf{B}_2 \\ \hline \mathbf{C}_2 & \mathbf{0} \end{array} \right)$$

Which defines substitutions:

$$\mathbf{A}_{n+1} = \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_n \\ \mathbf{C}_n & \mathbf{0} \end{pmatrix} \quad ; \quad \mathbf{B}_{n+1} = \begin{pmatrix} \mathbf{A}_n & \mathbf{0} \\ \mathbf{C}_n & \mathbf{0} \end{pmatrix}$$

$$\mathbf{C}_{n+1} = \begin{pmatrix} \mathbf{A}_n & \mathbf{B}_n \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

The lemma tells us that we could have started with the non-binary, non-negative matrix:

$$M_2 = \left(\begin{array}{cc|cc} 1 & 2 & 3 & 0 \\ 4 & 0 & 5 & 0 \\ \hline 6 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) = \left(\begin{array}{c|c} \widehat{\mathbf{A}}_2 & \widehat{\mathbf{B}}_2 \\ \hline \widehat{\mathbf{C}}_2 & \mathbf{0} \end{array} \right)$$

Where we are using "hats" ($\widehat{\mathbf{A}}_2$, etc.) to distinguish between the two sets of submatrices.

The indicator matrix of M_2 is the matrix V_2 given above for the Hard Square system.

Applying the V_2 substitution to M_2 gives us:

$$M_3 = \left(\begin{array}{cccc|cccc} 1 & 2 & 3 & 0 & 1 & 2 & 0 & 0 \\ 4 & 0 & 5 & 0 & 4 & 0 & 0 & 0 \\ 6 & 7 & 0 & 0 & 6 & 7 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) = \left(\begin{array}{cc|cc} \widehat{\mathbf{A}}_2 & \widehat{\mathbf{B}}_2 & \widehat{\mathbf{A}}_2 & \mathbf{0} \\ \widehat{\mathbf{C}}_2 & \mathbf{0} & \widehat{\mathbf{C}}_2 & \mathbf{0} \\ \hline \widehat{\mathbf{A}}_2 & \widehat{\mathbf{B}}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array} \right) = \left(\begin{array}{c|c} \widehat{\mathbf{A}}_3 & \widehat{\mathbf{B}}_3 \\ \hline \widehat{\mathbf{C}}_3 & \mathbf{0} \end{array} \right)$$

The lemma tells us that

$$\lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho (M_k)] = \lim_{k \rightarrow \infty} \frac{1}{k} \log [\rho (V_k)]$$

And thus, we can use the non-binary matrices, M_n , to compute the entropy of the Hard Square system. In other words, we can "re-weight" the V_2 matrix any way we like, as long as we preserve the position of the zero entries.

Our goal will be to re-weight the V_2 matrix in a way which will allow us to better approximate the entropy for the system.

Our initial search for a helpful re-weighting of V_2 begins by noting that the matrix substitutions themselves have an associated substitution matrix. Since each sub-matrix, A_n , is replaced by an A_n , a B_n and a C_n , each B_n is replaced by an A_n and a C_n , and each C_n is replaced by an A_n and a B_n , we see that the substitution matrix, S , is given by:

$$S = \begin{array}{c|ccc} & A_n & B_n & C_n \\ \hline A_n & 1 & 1 & 1 \\ B_n & 1 & 0 & 1 \\ C_n & 1 & 1 & 0 \end{array} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

If we rename the partition scheme of V_2 as:

$$V_2 = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) = \left(\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array} \right)$$

Then we know that the matrix D_k is replaced by only zeros at each stage of the substitution. We would then have the (familiar-looking) substitution matrix:

$$S = \begin{array}{c|cccc} & A_n & B_n & C_n & D_n \\ \hline A_n & 1 & 1 & 1 & 0 \\ B_n & 1 & 0 & 1 & 0 \\ C_n & 1 & 1 & 0 & 0 \\ D_n & 0 & 0 & 0 & 0 \end{array}$$

In fact, it is easy to see that:

Lemma 6.2 *If V_2 is the 2-wide vertical transition matrix and H_2 is the 2-tall horizontal transition matrix for a \mathbb{Z}^2 -action, \mathcal{G} , and if the matrix S is the substitution matrix which is defined by V_2 , then:*

$$\Psi(V_2) = H_2 = S$$

Where Ψ is the reflection operator defined in chapter 4.

Proof: This is a direct result of Lemma 1.

■

We know that the limiting behavior of the substitution algorithm can be understood by finding the Perron eigenvector of S , in the sense that the relative frequencies of the of the substitution blocks are given by the L_1 -normalized entries of the Perron eigenvector.

Let's examine the \mathbb{Z}^2 -action \mathcal{S}_h which we defined earlier.

This system has alphabet:

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

And forbidden set:

$$\mathcal{F} = \{ \boxed{1 \ 1} \}$$

The corresponding matrices for this system are given by:

$$V_2 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{A}_2 & \mathbf{B}_2 \\ \hline \mathbf{C}_2 & \mathbf{D}_2 \end{array} \right]$$

And:

$$S = H_2 = \Psi[V_2] = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \right]$$

It is interesting to note that the matrix H_2 is created by taking the Kronecker product of the standard fibonacci matrix with itself. Using the appropriate substitutions on H_2 is the equivalent of taking further Kronecker products. This will always be the case when the \mathbb{Z}^2 -action is created from vertically independent strips of a one-dimensional shift.

Continuing, we can see that the substitutions to create V_n are given by:

$$A_{n+1} = \begin{pmatrix} A_n & B_n \\ C_n & D_n \end{pmatrix} \quad ; \quad B_{n+1} = \begin{pmatrix} A_n & 0 \\ C_n & 0 \end{pmatrix}$$

$$C_{n+1} = \begin{pmatrix} A_n & B_n \\ 0 & 0 \end{pmatrix} \quad ; \quad D_{n+1} = \begin{pmatrix} A_n & 0 \\ 0 & 0 \end{pmatrix}$$

The L_1 normalized Perron eigenvector of H_2 is found to be:

$$\vec{v}_{H_2} = \frac{1}{\phi^2 + \phi + \phi + 1} \begin{pmatrix} \phi^2 \\ \phi \\ \phi \\ 1 \end{pmatrix} = \frac{1}{\phi^4} \begin{pmatrix} \phi^2 \\ \phi \\ \phi \\ 1 \end{pmatrix} = \begin{pmatrix} \phi^{-2} \\ \phi^{-3} \\ \phi^{-3} \\ \phi^{-4} \end{pmatrix}$$

Where ϕ represents the golden ratio, $\phi = \frac{1}{2}(1 + \sqrt{5})$.

This vector represents the relative frequencies of the substitution blocks as:

$$\begin{matrix} A_n \\ B_n \\ C_n \\ D_n \end{matrix} \begin{pmatrix} \phi^{-2} \\ \phi^{-3} \\ \phi^{-3} \\ \phi^{-4} \end{pmatrix}$$

We define the *quadrant weights* to be:

$$\widetilde{\overline{XY}}_k = \left(\sum_{\alpha, \beta \in \mathcal{A}^{k-1}} (V_k)_{[\alpha, \beta]} \right)$$

We define the *normalized quadrant weights* to be:

$$\widehat{\overline{XY}}_k = \left(\sum_{\alpha, \beta \in \mathcal{A}^{k-1}} (V_k)_{[\alpha, \beta]} \right) \left(\sum_{\alpha, \beta \in \mathcal{A}^k} (V_k)_{[\alpha, \beta]} \right)^{-1}$$

And define the *limiting normalized quadrant weights* to be:

$$\widehat{\overline{XY}} = \lim_{k \rightarrow \infty} \widehat{\overline{XY}}_k$$

From lemma 2, we know this limit exists, is finite, and is equal to the XY entry of the L_1 -normalized Perron eigenvector of H_2 . And thus, we can see that the limiting normalized quadrant weights of the V_k will be:

$$\left[\begin{array}{c|c} \widehat{00} & \widehat{01} \\ \hline \widehat{10} & \widehat{11} \end{array} \right] = \left[\begin{array}{c|c} \phi^{-2} & \phi^{-3} \\ \hline \phi^{-3} & \phi^{-4} \end{array} \right]$$

We can in fact stabilize the substitution process by weighting our original V_2 matrix according to the Perron eigenvector of H_2 . By re-weighting V_2 in this way, the quadrant masses will grow exactly according to the Perron eigenvalue of H_2 .

In this example, A_n should have a weight of ϕ^{-2} , similarly for B_n , etc.

This leads us to define the *stabilized initial matrix*, \tilde{V}_2 :

$$\tilde{V}_2 = \left(\begin{array}{cc|cc} \phi^{-2} & \phi^{-3} & \phi^{-2} & 0 \\ \phi^{-3} & \phi^{-4} & \phi^{-3} & 0 \\ \hline \phi^{-2} & \phi^{-3} & \phi^{-2} & 0 \\ 0 & 0 & 0 & 0 \end{array} \right)$$

Which has quadrant masses:

$$\left[\begin{array}{c|c} \widehat{00}_2 & \widehat{01}_2 \\ \hline \widehat{10}_2 & \widehat{11}_2 \end{array} \right] = \left[\begin{array}{c|c} 1 & \phi^{-1} \\ \hline \phi^{-1} & \phi^{-2} \end{array} \right] = \phi^2 \left[\begin{array}{c|c} \phi^{-2} & \phi^{-3} \\ \hline \phi^{-3} & \phi^{-4} \end{array} \right]$$

Exactly as in the limiting case. If we substitute, we obtain the matrix:

$$\tilde{V}_3 = \left(\begin{array}{cccc|cccc} \phi^{-2} & \phi^{-3} & \phi^{-2} & 0 & \phi^{-2} & \phi^{-3} & 0 & 0 \\ \phi^{-3} & \phi^{-4} & \phi^{-3} & 0 & \phi^{-3} & \phi^{-4} & 0 & 0 \\ \phi^{-2} & \phi^{-3} & \phi^{-2} & 0 & \phi^{-2} & \phi^{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \phi^{-2} & \phi^{-3} & \phi^{-2} & 0 & \phi^{-2} & \phi^{-3} & 0 & 0 \\ \phi^{-3} & \phi^{-4} & \phi^{-3} & 0 & \phi^{-3} & \phi^{-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

Which has quadrant masses:

$$\left[\begin{array}{c|c} \widehat{00}_3 & \widehat{01}_3 \\ \hline \widehat{10}_3 & \widehat{11}_3 \end{array} \right] = \left[\begin{array}{c|c} \phi^2 & \phi \\ \hline \phi & 1 \end{array} \right] = \phi^4 \left[\begin{array}{c|c} \phi^{-2} & \phi^{-3} \\ \hline \phi^{-3} & \phi^{-4} \end{array} \right]$$

Which is as we expect, given that the Perron eigenvalue of H_2 is ϕ^2 . Further substitutions will have the same quadrant weights, each time multiplied by ϕ^2 , and thus they will all have the same normalized quadrant weights.

We can make the same type of stabilization for any \mathbb{Z}^2 -action.

Of particular interest to us is the Hard Square model, which has:

$$V_2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} ; \quad \vec{v}_{H_2} = \begin{matrix} A_n \\ B_n \\ C_n \\ D_n \end{matrix} \begin{pmatrix} \sqrt{2} \\ 1 \\ 1 \\ 0 \end{pmatrix} \frac{1}{2 + \sqrt{2}} ; \quad \rho(H_2) = 1 + \sqrt{2}$$

Re-weighting V_2 , we obtain the matrix:

$$\tilde{V}_2 = \begin{pmatrix} \sqrt{2} & 1 & \sqrt{2} & 0 \\ 1 & 0 & 1 & 0 \\ \sqrt{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

And we see that the quadrant masses for \tilde{V}_k will be:

$$\left[\begin{array}{c|c} \widehat{00}_k & \widehat{01}_k \\ \hline \widehat{10}_k & \widehat{11}_k \end{array} \right] = (1 + \sqrt{2})^{k-2} \left[\begin{array}{c|c} 2 + \sqrt{2} & 1 + \sqrt{2} \\ \hline 1 + \sqrt{2} & 0 \end{array} \right]$$

Lemma 1 tells us that re-weighting the matrices in this manner and computing the Perron eigenvalues for the \tilde{V}_k will still yield the correct entropy for the system in the sense that:

$$h = \lim_{n \rightarrow \infty} \frac{1}{n} \log \rho(V_n) = \lim_{n \rightarrow \infty} \frac{1}{n} \log \rho(\tilde{V}_n)$$

Estimating the entropy by the $\rho(\tilde{V}_n)$ in the equation above does give a slightly better approximation than by using the $\rho(V_n)$, but the improvement is very small, and the rate of convergence to the true entropy remains the same.

7 Geometry of the V_n matrices and their eigenvectors

Now, we will change our focus to the substitutions themselves. With two-dimensional iterated substitutions like those defined in chapter 3, we expect to get some fractal geometry.

We begin by defining a map, \mathcal{M} , which takes $n \times n$ non-negative matrices to subsets of the unit square. Let T be an $n \times n$ non-negative matrix, and define the set \mathcal{P} as follows:

$$\mathcal{P} = \{(i, j) : T_{[i,j]} > 0\}$$

Then define $\mathcal{M}(T)$ by:

$$\mathcal{M}(T) = \bigcup_{(i,j) \in \mathcal{P}} \left\{ x \in [0, 1) : \frac{i-1}{n} \leq x < \frac{i}{n} \right\} \times \left\{ y \in [0, 1) : \frac{j-1}{n} \leq y < \frac{j}{n} \right\}$$

For example:

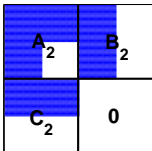
$$\mathcal{M} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \text{[Diagram: A square with a white square in the bottom-right corner, representing the set defined by the matrix.]}$$

And:

$$\mathcal{M} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \text{[Diagram: A square with a white square in the center, representing the set defined by the matrix.]}$$

In each set, we are designating the top-left corner the origin (0,0) for the unit square, and the bottom right is the point (1,1). This will allow us to "see" the matrix better as a set.

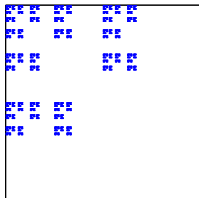
Now, we can partition the unit square exactly as our matrices are partitioned for the substitution algorithm. For the Hard Square system:

$$\mathcal{M}(V_2) = \mathcal{M} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \mathcal{M} \left(\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & 0 \end{array} \right) =$$


We can see that substitution on matrices is equivalent to a union of contraction maps on the unit square. Thus, for any V_2 matrix, there will be a **limiting fractal**, which we call $\mathcal{M}_\infty(V_2)$, defined as:

$$\mathcal{M}_\infty(V_2) = \lim_{k \rightarrow \infty} \mathcal{M}(V_k)$$

For the Hard Square system, the limiting fractal looks like:

$$\mathcal{M}_\infty(V_2) = \mathcal{M}_\infty \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} =$$


Note that the Hausdorff dimension of the limiting fractal will not be explored in this paper, but its uses are being explored.

For the two systems given by horizontally or vertically independent Fibonacci shifts described earlier:

$$\mathcal{M}_\infty \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{img alt="A square fractal pattern consisting of a 3x3 grid of smaller squares, each containing a 3x3 grid of even smaller squares, forming a Sierpinski gasket-like structure." data-bbox="541 158 658 248"/>$$

$$\mathcal{M}_\infty \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{img alt="A square fractal pattern consisting of a 3x3 grid of smaller squares, each containing a 3x3 grid of even smaller squares, forming a Sierpinski gasket-like structure." data-bbox="541 261 658 351"/>$$

It is interesting to note that the limiting fractal for the horizontally independent system of Fibonacci shifts, \mathcal{S}_v , is the familiar *Sierpinski Gasket* fractal.

Now we will define a map, \mathcal{D} which takes a non-negative, non-zero, L_1 normalized vector to a function on the unit interval. Let \vec{v} be an $n \times 1$ vector with non-negative entries which sum to one. Define the distribution function $\mathcal{D}_{\vec{v}}(x)$ on the unit interval $(0,1]$ by:

$$\mathcal{D}_{\vec{v}}(x) = \sum_{i=1}^j \vec{v}_{[i,1]} \quad \text{when} \quad \frac{j-1}{n} \leq x < \frac{j}{n} \quad \text{for} \quad j = 1 \dots n$$

We can see that since \vec{v} is non-negative and non-zero, $\mathcal{D}_{\vec{v}}(x)$ is monotone increasing on the unit interval, and that $\mathcal{D}_{\vec{v}}(1) = 1$. That is, we are creating the cumulative distribution function for the vector in the usual manner [3], from which we state a theorem, and add several corollaries to the current context.

Theorem 7.1 *Let V_2 be given by a non-empty \mathbb{Z}^2 -action and create V_n by*

applying the corresponding substitutions. Let \vec{v}_n denote the Perron eigenvector of V_n .

Then the functions $\mathcal{D}_{\vec{v}_n}(x)$ converge uniformly to a function $\mathcal{D}_{\vec{V}_2}^\infty(x)$ on the unit interval.

Corollary 7.1 *If we eliminate the zero rows and columns of the V_n (which we called \check{V}_n in chapter 3), then the corresponding $\mathcal{D}_{\vec{v}_n}(x)$ will converge to a strictly monotone increasing function $\mathcal{D}_{\vec{V}_2}^\infty(x)$ on the unit interval.*

Corollary 7.2 *Let M be an $n^2 \times n^2$ non-negative matrix and $V_2 = \mathbb{I}(M)$. Suppose V_2 defines a valid \mathbb{Z}^2 -action. Let M_{k+2} be the matrix obtained by applying the V_2 substitutions k times to the matrix M . Let \vec{v}_k and \vec{m}_k denote the L_1 normalized Perron eigenvectors of V_k and M_k respectively. Then:*

$$\lim_{k \rightarrow \infty} \mathcal{D}_{\vec{v}_k}(x) = \lim_{k \rightarrow \infty} \mathcal{D}_{\vec{m}_k}(x) \quad \forall x \in [0, 1]$$

Corollary 7.3 *If a \mathbb{Z}^2 -action, \mathcal{G} , exhibits horizontal or vertical independence, and \vec{v}_n are the Perron eigenvectors of the associated transfer matrices, V_n , then the distributions $\mathcal{D}_{\vec{v}_n}(x)$ form a Martingale with respect to the σ -fields generated by the intervals $\left\{ \left[\frac{i}{|\mathcal{A}|^n}, \frac{i+1}{|\mathcal{A}|^n} \right] : i = 0 \cdots |\mathcal{A}|^n - 1 \right\}$.*

We can see (with numerical experiments) that by stabilizing the V_2 matrix (as before), the corresponding distributions converge slightly faster to the limiting distribution, however the improvement is small.

For the Hard Square \mathbb{Z}^2 -action we can visualize the limiting distribution:

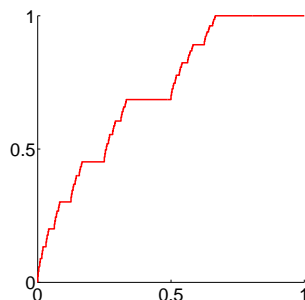


Figure 4: $\mathcal{D}_{V_2}^\infty(x)$ for the Hard Square system

At first glance, one may think that this distribution exhibits self-similarity between the intervals $[0, 1/2]$ and $[1/2, 1]$. However, it is not self-similar.

If we eliminate the zero rows and columns for the Hard Square system, We see:

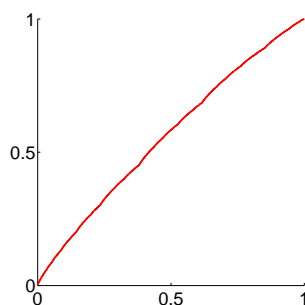


Figure 5: $\mathcal{D}_{V_2}^\infty(x)$ for the Hard Square system

It is visually informative (and aesthetically pleasing) to view the limiting fractal and the limiting distribution function together. To get a better visual interpretation of their relationship, however, we maintain that the origin be at the top left for the limiting fractal, but at the bottom left for the limiting distribution. That is, we are simply super-imposing the two images.

For the Hard Square system, we can see (within the bounds of resolution):

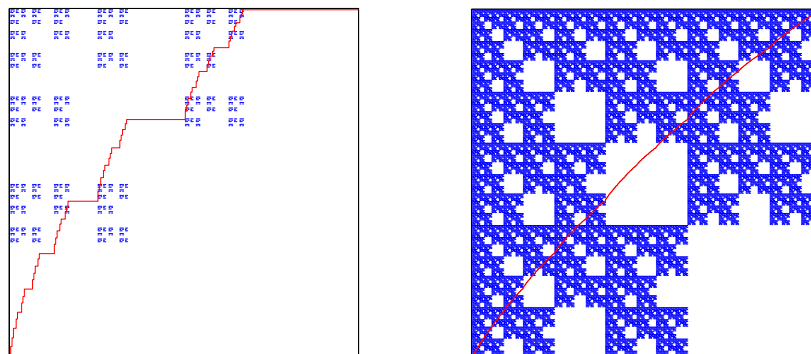


Figure 6: $\mathcal{D}_{V_2}^\infty(x)$ and $\mathcal{D}_{V_2}^\infty(x)$ for the Hard Square system, together with their limiting fractals

Note that by eliminating the zero rows and columns of our matrices, Lebesgue measure no longer applies in the sense that our (now infinite) alphabet, $\mathcal{A}^{\mathbb{Z}^+}$, covers the entire unit interval and thus has measure 1. This is part of our initial motivation for keeping forbidden letters in our initial matrices.

When we look at the vertically independent Fibonacci strips, \mathcal{S}_h , we get:

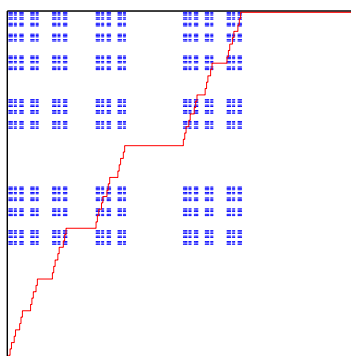


Figure 7: $\mathcal{D}_{V_2}^\infty(x)$ for \mathcal{S}_h , together with its limiting fractal

Note that for the system \mathcal{S}_h , if we eliminate the zero rows and columns, the

limiting fractal has the entire unit square, $[0, 1) \times [0, 1)$ for support and

$\mathcal{D}_{\tilde{V}_2}^\infty(x) = x$ (again, we can no longer apply Lebesgue measure).

It is easy to see that this must always be the case when a system consists of vertically independent shift systems.

We also note that this distribution exhibits a self-similar structure between the intervals $[0, 1/2]$ and $[1/2, 1]$. We can then conclude that this is the measure of a Bernoulli process [3]. This will always be the case when a \mathbb{Z}^d -action exhibits vertical independence.

When we look at the horizontally independent Fibonacci strips, \mathcal{S}_v , we see:

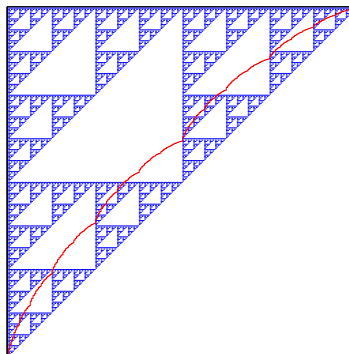


Figure 8: $\mathcal{D}_{\tilde{V}_2}^\infty(x)$ for \mathcal{S}_v , together with its limiting fractal

In this case, we note that there are no non-zero rows or columns. Thus,

$\mathcal{D}_{\tilde{V}_2}^\infty(x) = \mathcal{D}_{\tilde{V}_2}^\infty(x)$. This will always be true when a system consists of horizontally independent shift systems.

Furthermore, this distribution is self-similar between the intervals $[0, 1/2]$ and

$[1/2, 1]$, and thus this is the measure of a Bernoulli process [3]. This will always be the case when a \mathbb{Z}^d -action exhibits horizontal independence.

Also note that for each system, \mathcal{S}_h and \mathcal{S}_v , the exact entropy is the golden ratio, $\frac{1}{2}(1 + \sqrt{5})$. Thus, in the case of \mathcal{S}_v , there is a link between the golden ratio and the Sierpinski Gasket. (At this time, the author does not know of any other correlation between the two, and is truly amazed.)

Some other interesting examples to note are the *domino* \mathbb{Z}^2 -action which has alphabet:

$$\mathcal{A} = \{ \boxed{0}, \boxed{1}, \boxed{2}, \boxed{3} \}$$

And forbidden set:

$$\mathcal{F} = \left\{ \begin{array}{c} \boxed{0} \\ \boxed{0} \end{array}, \begin{array}{c} \boxed{1} \\ \boxed{1} \end{array}, \begin{array}{c} \boxed{1} \\ \boxed{2} \end{array}, \begin{array}{c} \boxed{1} \\ \boxed{3} \end{array}, \begin{array}{c} \boxed{2} \\ \boxed{0} \end{array}, \begin{array}{c} \boxed{3} \\ \boxed{0} \end{array}, \right. \\ \left. \begin{array}{c} \boxed{0} \boxed{3} \end{array}, \begin{array}{c} \boxed{1} \boxed{3} \end{array}, \begin{array}{c} \boxed{2} \boxed{0} \end{array}, \begin{array}{c} \boxed{2} \boxed{1} \end{array}, \begin{array}{c} \boxed{2} \boxed{2} \end{array}, \begin{array}{c} \boxed{3} \boxed{3} \end{array} \right\}$$

Note that the forbidden set forces the open sides of the alphabet letters to match up.

The domino system has initial transfer matrix:

$$V_2 = \begin{array}{c} 00 \\ 01 \\ 02 \\ 03 \\ \hline 10 \\ 11 \\ 12 \\ 13 \\ \hline 20 \\ 21 \\ 22 \\ 23 \\ \hline 30 \\ 31 \\ 32 \\ 33 \end{array} \left(\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

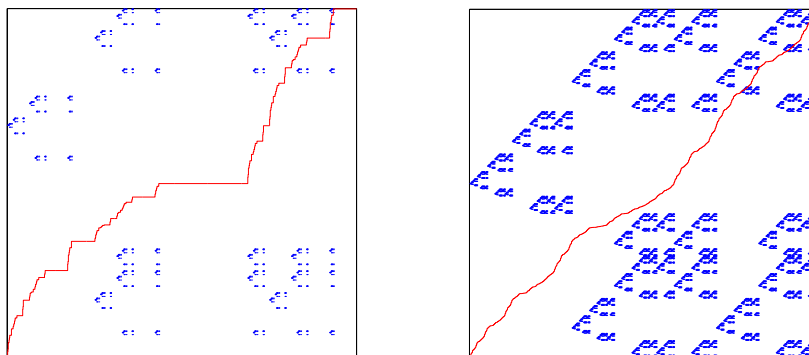


Figure 9: $\mathcal{D}_{V_2}^\infty(x)$ and $\mathcal{D}_{V_2}^\infty(x)$ for the Domino system, together with their limiting fractals

One can see that this system describes the arrangements for dominos on a grid. The entropy for the domino system was computed in [5] and is known to be the Catalan constant.

Another important example is the famous *Square Ice* or *Six Vertex* model.

The entropy for this system was solved exactly in [14], and is known to be

$(4/3) \log(3/2)$. The alphabet size is 6, and thus the forbidden set and initial transfer matrix are quite large. Without explicitly stating them, we see:

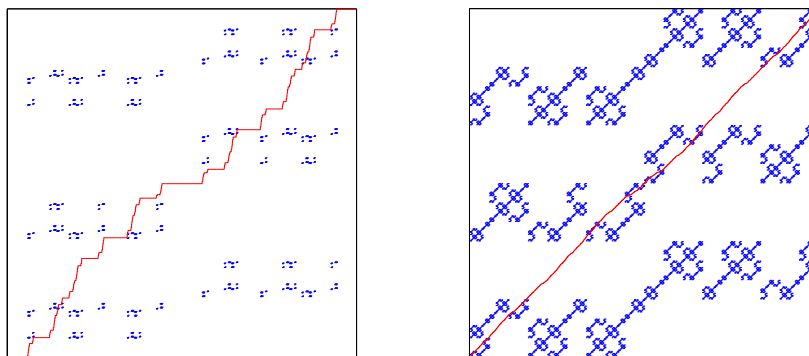


Figure 10: $\mathcal{D}_{V_2}^\infty(x)$ and $\mathcal{D}_{V_2}^\infty(x)$ for the Six Vertex Model, together with their limiting fractals

The square ice model is known to be equivalent to the *3-coloring model*, that is to say that there is a one to one map between the two systems. The 3-coloring model has an alphabet size of 3, with the condition that no tile can share an edge with a tile of the same name. That is:

$$\mathcal{A} = \{ \boxed{0}, \boxed{1}, \boxed{2} \}$$

$$\mathcal{F} = \left\{ \begin{array}{ccc} \boxed{0} & \boxed{1} & \boxed{2} \\ \boxed{0} & \boxed{1} & \boxed{2} \\ \boxed{0\ 0} & \boxed{1\ 1} & \boxed{2\ 2} \end{array} \right\}$$

The 3-coloring system has initial 2-wide transfer matrix:

$$V_2 = \begin{array}{c} 00 \\ 01 \\ 02 \\ \hline 10 \\ 11 \\ 12 \\ \hline 20 \\ 21 \\ 22 \end{array} \begin{pmatrix} 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0 \\ 0 & 0 & 0 & | & 1 & 0 & 1 & | & 1 & 0 & 0 \\ 0 & 0 & 0 & | & 1 & 0 & 0 & | & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & | & 0 & 0 & 0 & | & 0 & 1 & 0 \\ 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0 \\ 0 & 1 & 0 & | & 0 & 0 & 0 & | & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & | & 0 & 0 & 1 & | & 0 & 0 & 0 \\ 0 & 0 & 1 & | & 1 & 0 & 1 & | & 0 & 0 & 0 \\ 0 & 0 & 0 & | & 0 & 0 & 0 & | & 0 & 0 & 0 \end{pmatrix}$$

$$\check{V}_2 = \begin{array}{c} 01 \\ 02 \\ \hline 10 \\ 12 \\ \hline 20 \\ 21 \end{array} \begin{pmatrix} 0 & 0 & | & 1 & 1 & | & 1 & 0 \\ 0 & 0 & | & 1 & 0 & | & 1 & 1 \\ \hline 1 & 1 & | & 0 & 0 & | & 0 & 1 \\ 1 & 0 & | & 0 & 0 & | & 1 & 1 \\ \hline 1 & 1 & | & 0 & 1 & | & 0 & 0 \\ 0 & 1 & | & 1 & 1 & | & 0 & 0 \end{pmatrix}$$

It is interesting to note that the row sums of initial transfer matrix, V_2 and its substitutions, V_n , have strong (yet unknown) connection to the Farey fractions.

For the 3-coloring system, we see:

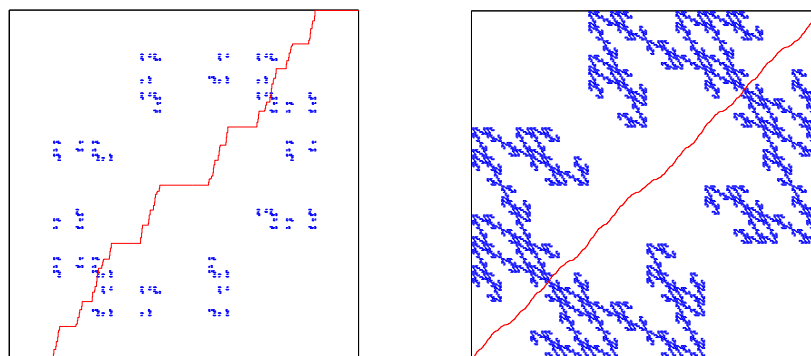


Figure 11: $D_{V_2}^\infty(x)$ and $D_{\check{V}_2}^\infty(x)$ for the 3-coloring model, together with their limiting fractals

8 Algebraic manipulations of the substitution algorithm

In this section, we will attempt to use the eigenvectors of the matrices, V_n (generated by the substitution algorithm in section 3), to approximate the entropy for a \mathbb{Z}^2 -action. We do this by exploiting the self-similar aspects of the matrices, V_n .

We will begin by examining a simple \mathbb{Z}^2 -action, and then trying to mimic what we find and try to apply those discoveries to more general actions.

The system that we will begin with is the \mathbb{Z}^2 -action, \mathcal{S}_h (introduced in section 6), which consists of an infinite stack of horizontal 1-dimensional Fibonacci-shifts, with no vertical correlations.

This system has alphabet:

$$\mathcal{A} = \{ \boxed{0} , \boxed{1} \}$$

And forbidden set:

$$\mathcal{F} = \{ \boxed{1 \ 1} \}$$

This system has its initial substitution matrix (as in chapter 6) as:

$$V_2 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array} \right]$$

And we can see that the substitutions to create V_n are given by:

$$A_{n+1} = \begin{pmatrix} A_n & B_n \\ C_n & D_n \end{pmatrix} \quad ; \quad B_{n+1} = \begin{pmatrix} A_n & 0 \\ C_n & 0 \end{pmatrix}$$

$$C_{n+1} = \begin{pmatrix} A_n & B_n \\ 0 & 0 \end{pmatrix} \quad ; \quad D_{n+1} = \begin{pmatrix} A_n & 0 \\ 0 & 0 \end{pmatrix}$$

We will denote by λ_n the Perron eigenvalue of V_n , and we will use \vec{v}_n to denote the corresponding Perron eigenvector, the sum of whose entries is one. (That is, the L_1 -unit Perron eigenvector).

We will partition the Perron eigenvector, \vec{v}_n , according to the alphabet, \mathcal{A} , as follows:

$$\vec{v}_n = \begin{pmatrix} X_n \\ Y_n \end{pmatrix}$$

At this point, we will make an educated assumption about the nature of \vec{v}_n .

We'll suggest that:

$$\vec{v}_n = \begin{pmatrix} X_n \\ Y_n \end{pmatrix} = \begin{pmatrix} \|X_n\| \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix} \\ \frac{\|Y_n\|}{\|X_{n-1}\|} \begin{pmatrix} X_{n-1} \\ 0 \end{pmatrix} \end{pmatrix}$$

That is to say, that we are suggesting a possible substitution algorithm for the Perron eigenvector itself. The norms are included in the eigenvector substitution to ensure that the L_1 -norm remains unchanged. We will now test to see if our eigenvector substitution works.

To test this, we begin with the standard eigenvector equation, which we will then expand using the substitution algorithm and the partitioned \vec{v}_n .

$$V_n \vec{v}_n = \lambda_n \vec{v}_n$$

$$\implies \begin{bmatrix} A_n & B_n \\ C_n & D_n \end{bmatrix} \begin{pmatrix} X_n \\ Y_n \end{pmatrix} = \lambda_n \begin{pmatrix} X_n \\ Y_n \end{pmatrix}$$

From the first row of this equation:

$$A_n X_n + B_n Y_n = \lambda_n X_n$$

Using the matrix and eigenvector substitutions, we get:

$$\begin{pmatrix} A_{n-1} & B_{n-1} \\ C_{n-1} & D_{n-1} \end{pmatrix} \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix} \|X_n\| + \begin{pmatrix} A_{n-1} & 0 \\ C_{n-1} & 0 \end{pmatrix} \begin{pmatrix} X_{n-1} \\ 0 \end{pmatrix} \frac{\|Y_n\|}{\|X_{n-1}\|} = \lambda_n \|X_n\| \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix}$$

We now see that the first term in this expression is $V_{n-1} \vec{v}_{n-1} \|X_n\|$. Thus we have:

$$\lambda_{n-1} \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix} \|X_n\| + \begin{pmatrix} A_{n-1} X_{n-1} \\ C_{n-1} X_{n-1} \end{pmatrix} \frac{\|Y_n\|}{\|X_{n-1}\|} = \lambda_n \|X_n\| \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix}$$

The first row of this equation gives us:

$$\lambda_{n-1} \|X_n\| X_{n-1} + A_{n-1} X_{n-1} \frac{\|Y_n\|}{\|X_{n-1}\|} = \lambda_n \|X_n\| X_{n-1}$$

Substituting again gives us:

$$\begin{aligned} \lambda_{n-1} \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\| \begin{pmatrix} \mathbf{X}_{n-2} \\ \mathbf{Y}_{n-2} \end{pmatrix} + \begin{pmatrix} \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{C}_{n-2} & \mathbf{D}_{n-2} \end{pmatrix} \|\mathbf{X}_{n-1}\| \begin{pmatrix} \mathbf{X}_{n-2} \\ \mathbf{Y}_{n-2} \end{pmatrix} \frac{\|\mathbf{Y}_n\|}{\|\mathbf{X}_{n-1}\|} \\ = \lambda_n \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\| \begin{pmatrix} \mathbf{X}_{n-2} \\ \mathbf{Y}_{n-2} \end{pmatrix} \end{aligned}$$

The second term of this equation now satisfies an eigenvector equation. Thus, by cancelling the $\|\mathbf{X}_n\|$ in the second term, the first row of this equation yields:

$$\lambda_{n-1} \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\| \mathbf{X}_{n-2} + \|\mathbf{Y}_n\| \lambda_{n-2} \mathbf{X}_{n-2} = \lambda_n \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\| \mathbf{X}_{n-2}$$

At this point, we do not need to substitute any further, since every term has is now expressed only in terms of eigenvalues and eigenvectors. We have at this point:

$$\lambda_{n-1} \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\| + \|\mathbf{Y}_n\| \lambda_{n-2} = \lambda_n \|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\|$$

Which we write as:

$$\lambda_n = \lambda_{n-1} + \frac{\|\mathbf{Y}_n\|}{\|\mathbf{X}_n\| \cdot \|\mathbf{X}_{n-1}\|} \lambda_{n-2} \quad (7)$$

We have successfully written the eigenvalue, λ_n as a recursion of previous eigenvalues!

If the coefficient of λ_{n-2} in the above equation converges with n , then we can use the limiting recursion to obtain the true entropy for the system.

With this particular system, we can cheat a bit since it is easy to see that the eigenvectors are generated by the Fibonacci sequence. That is:

$$\begin{pmatrix} \|X_2\| \\ \|Y_2\| \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 2 \\ 1 \end{pmatrix} ; \quad \begin{pmatrix} \|X_3\| \\ \|Y_3\| \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 3 \\ 2 \end{pmatrix} ; \quad \begin{pmatrix} \|X_4\| \\ \|Y_4\| \end{pmatrix} = \frac{1}{8} \begin{pmatrix} 5 \\ 3 \end{pmatrix} ; \quad \dots$$

From this it is easy to show that for every n , we have:

$$\frac{\|Y_n\|}{\|X_n\| \cdot \|X_{n-1}\|} = 1$$

And thus our equation becomes the familiar Fibonacci recursion:

$$\lambda_n = \lambda_{n-1} + \lambda_{n-2}$$

From which we can deduce that:

$$\lim_{n \rightarrow \infty} (\lambda_n)^{1/n} = \frac{1 + \sqrt{5}}{2}$$

Which is indeed the correct entropy for this system.

At this point it is important to note that at several steps in the above calculation, we chose a certain row of a matrix equation as our focus.

It turns out that in this calculation, the choice of row at each step does not affect the end result.

We now attempt the same style of manipulation on the hard square model, which has:

$$\mathcal{A} = \{ \boxed{0}, \boxed{1} \} \quad ; \quad \mathcal{F} = \{ \boxed{11}, \boxed{\begin{array}{c} 1 \\ 1 \end{array}} \}$$

This system has the initial substitution matrix :

$$V_2 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & 0 \end{array} \right]$$

With substitutions given by:

$$A_{n+1} = \begin{pmatrix} A_n & B_n \\ C_n & 0 \end{pmatrix} \quad ; \quad B_{n+1} = \begin{pmatrix} A_n & 0 \\ C_n & 0 \end{pmatrix}$$

$$C_{n+1} = \begin{pmatrix} A_n & B_n \\ 0 & 0 \end{pmatrix}$$

Again, we partition the Perron eigenvector, \vec{v}_n , as:

$$\vec{v}_n = \begin{pmatrix} X_n \\ Y_n \end{pmatrix}$$

We'll now make the unsupported assumption that the hard square eigenvectors behave like the system described earlier:

$$X_n = \|X_n\| \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix}$$

And:

$$Y_n = \frac{\|Y_n\|}{\|X_{n-1}\|} \begin{pmatrix} X_{n-1} \\ 0 \end{pmatrix}$$

We note now that this assumption is not true. However, it is not terribly wrong. If we look at the distribution functions of the Hard Square and \mathcal{S}_h eigenvectors, we see:

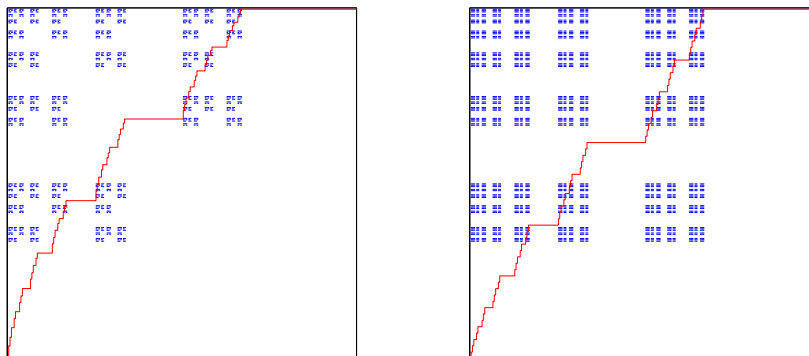


Figure 12: $\mathcal{D}_{V_2}^\infty(x)$ for the Hard Square and \mathcal{S}_h systems, together with their limiting fractals

The distributions look similar and are "close enough" in the sense that they share the same support and have only small differences in their function values. It turns out that this is enough to yield a surprisingly accurate estimate for the hard square entropy.

As before, we start with the eigenvector equation:

$$V_n \vec{v}_n = \begin{pmatrix} A_n & B_n \\ C_n & 0 \end{pmatrix} \begin{pmatrix} X_n \\ Y_n \end{pmatrix} = \begin{pmatrix} A_n X_n + B_n Y_n \\ C_n X_n \end{pmatrix} = \lambda_n \begin{pmatrix} X_n \\ Y_n \end{pmatrix}$$

The second row of this equation gives us:

$$C_n X_n = \begin{pmatrix} A_{n-1} & B_{n-1} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix} \|X_n\| = \lambda_n Y_n$$

The first row gives us:

$$\begin{pmatrix} A_{n-1}X_{n-1} + B_{n-1}Y_{n-1} \\ 0 \end{pmatrix} \|X_n\| = \lambda_{n-1} \begin{pmatrix} X_{n-1} \\ 0 \end{pmatrix} \|X_n\| = \lambda_n \frac{\|Y_n\|}{\|X_{n-1}\|} \begin{pmatrix} X_{n-1} \\ 0 \end{pmatrix}$$

Which yields:

$$\lambda_{n-1} \|X_n\| = \lambda_n \frac{\|Y_n\|}{\|X_{n-1}\|}$$

Using the fact that $\|Y_n\| = 1 - \|X_n\|$:

$$\frac{\lambda_n}{\lambda_{n-1}} = \frac{\|X_n\| \cdot \|X_{n-1}\|}{1 - \|X_n\|}$$

It is widely believed that the left side of this equation converges to the correct entropy. But the right side still depends on our naive assumption about the eigenvectors. However, the right side must converge to some number, since we know that the $\|X_n\|$ converge.

By using MATLAB to numerically estimate the values of the $\|X_n\|$, and assuming that for "large" n , $\|X_n\| \approx \|X_{n-1}\|$, we obtain the following computations:

n	$\frac{\ X_n\ ^2}{1-\ X_n\ } - h_B$
3	-0.0754409415805952
4	0.0194488210548154
5	-0.0182087948581906
6	-0.00320522050757299
7	-0.00940138070106267
8	-0.00681222368996282
9	-0.00791045720681782
10	-0.00744015843836565
11	-0.00764337288239991
12	-0.00755493525670037
13	-0.00759366757134727

Where h_B is the hard square entropy estimate given by Baxter in [1].

We note that the best estimate is for $n = 6$, and that the estimates are converging to an incorrect number which is still remarkably close to the correct entropy, given that we made such loose assumptions about the behavior of the eigenvectors.

One hope is that we might combine this technique with the re-weighting methods of chapter 6 in order to get better approximations to the correct entropy.

9 Numerical Estimations for the Entropy of \mathbb{Z}^2 -actions

In this chapter, we will construct and explore a new numerical method for computing the entropy of a \mathbb{Z}^2 -action. We construct this method by making analogies with one-dimensional shifts of finite type, which we analyze first.

At the time of submission for this thesis, a proof of convergence for this numerical estimator has not been found.

Let \mathcal{G} be a one-dimensional shift of finite type with alphabet \mathcal{A} and finite forbidden set \mathcal{F} . Then there is an associated graph and transfer matrix, $T_{\mathcal{G}}$, that generates \mathcal{G} . The number of allowed n -blocks in \mathcal{G} can be found by finding $\|T_{\mathcal{G}}^{n-1}\vec{\mathbf{1}}\|_1$ where $\vec{\mathbf{1}}$ is an $|\mathcal{A}| \times 1$ vector with a 1 in every entry. (Recall that $\|\vec{v}\|_1$ denotes the sum of the entries of the vector \vec{v} , and we don't need to worry about absolute values here, since every number is positive in our current setting.)

If we know the entire sequence of numbers $\{ |\mathcal{B}_n(\mathcal{G})| \}_{n=1}^{\infty}$, then we can approximate the entropy by computing $\frac{1}{n} \log |\mathcal{B}_n(\mathcal{G})|$ for large n . For an arbitrary system, we don't expect that $\frac{1}{n} \log |\mathcal{B}_n(\mathcal{G})| = h$ for any n . As stated previously, we expect that the sequence $\frac{1}{n} \log |\mathcal{B}_n(\mathcal{G})|$ will converge very slowly to the correct entropy. Thus, in order to get a good approximation of the entropy, we need a very large n , which is not always computationally practical.

However, we can use our knowledge of the beginning of the sequence

$\{ |\mathcal{B}_n(\mathcal{G})| \}_{n=1}^\infty$ to construct a series of matrices, A_n , which estimates the growth rate of $|\mathcal{B}_n(\mathcal{G})|$.

For the sake of brevity, we will now adopt the notation $b_k = |\mathcal{B}_k(\mathcal{G})|$, and $\vec{\mathbf{1}}_k$ is a $k \times 1$ vector with a 1 in every entry. We will also call $H = 2^h$, as this will be used frequently in this section.

We know that $b_k = \|T_{\mathcal{G}}^{k-1} \vec{\mathbf{1}}_{|\mathcal{A}|}\|_1$ and thus, we can think of the sequence $\{b_k\}_{k=1}^\infty$ as a sequence of norms of the orbit of the vector $\vec{\mathbf{1}}_{|\mathcal{A}|}$ under the operator $T_{\mathcal{G}}$. Furthermore, since $T_{\mathcal{G}}$ is a non-negative matrix, we know that the orbit of $\vec{\mathbf{1}}_{|\mathcal{A}|}$ lies entirely in the positive (first) quadrant of $\mathbb{R}^{|\mathcal{A}|}$.

In general, there are many matrices, T , that could produce orbits such that $\|T^{n-1} \vec{\mathbf{1}}_{|\mathcal{A}|}\|_1 = b_n$.

To help with this, we exploit the fact that topology does not distinguish between norms on \mathbb{R}^n . If an orbit has an asymptotic growth rate (determined by the spectral radius of the operator) in L_1 , then it has the same growth rate in L_∞ .

For every n , construct a sequence of $n + 1$ vectors as follows:

$$\vec{v}_{1,n} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad \vec{v}_{2,n} = \begin{pmatrix} b_2 \\ b_3 \\ \vdots \\ b_{n+1} \end{pmatrix} \quad \dots \quad \vec{v}_{n,n} = \begin{pmatrix} b_n \\ b_{n+1} \\ \vdots \\ b_{2n-1} \end{pmatrix} \quad \vec{v}_{n+1,n} = \begin{pmatrix} b_{n+1} \\ b_{n+2} \\ \vdots \\ b_{2n} \end{pmatrix}$$

The first index indicates the index of the first entry of the vector, the second indicates the length of the vector.

We first note that since the b_k are a monotone sequence, the sup norm of each vector is equal to its last entry. That is to say $\|v_{k,n}\|_\infty = b_{n+k-1}$, and we see that this sequence of vectors has the same growth rate in the sup norm.

We wish to find a sequence of matrices, A_n , with the property that $A_n \vec{v}_{k,n} = \vec{v}_{k+1,n}$ for every $1 \leq k \leq n$, so that A_n will approximate the actual growth rate of the b_n in the sense that the orbit of $\vec{\mathbf{1}}_{|\mathcal{A}|}$ will have the correct growth rate.

If the set of vectors, $\{\vec{v}_{1,n}, \vec{v}_{2,n}, \dots, \vec{v}_{n,n}\}$, is linearly independent, then they form a basis for \mathbb{R}^n , and $\vec{v}_{n+1,n}$ can be uniquely written as a linear combination of this set. Thus, there will be a unique solution to the system:

$$\begin{aligned}
 \alpha_{0,n}b_1 + \alpha_{1,n}b_2 + \alpha_{2,n}b_3 + \cdots + \alpha_{n-1,n}b_n &= b_{n+1} \\
 \alpha_{0,n}b_2 + \alpha_{1,n}b_3 + \alpha_{2,n}b_4 + \cdots + \alpha_{n-1,n}b_{n+1} &= b_{n+2} \\
 &\vdots \\
 \alpha_{0,n}b_n + \alpha_{1,n}b_{n+1} + \alpha_{2,n}b_{n+2} + \cdots + \alpha_{n-1,n}b_{2n-1} &= b_{2n}
 \end{aligned} \tag{8}$$

If $\{\vec{v}_{1,n}, \vec{v}_{2,n}, \dots, \vec{v}_{n,n}\}$ is not a linearly independent set of vectors, basic linear algebra states that there could be infinitely many solutions or no solution. We first make the assumption that the system has at least one solution.

We also wish to make the further assumption that If $\{\vec{v}_{1,n}, \vec{v}_{2,n}, \dots, \vec{v}_{n,n}\}$ is linearly dependent, then the set of vectors $\{\vec{v}_{1,n+1}, \vec{v}_{2,n+1}, \dots, \vec{v}_{n+1,n+1}\}$ is also linearly dependent. This is the assumption which tells us that once we have found a linearly dependent set, we have fully characterized the system.

Finding the α_k is as easy as row-reducing the augmented Hankel matrix:

$$\left[\begin{array}{cccc|c} \vec{v}_{1,n} & \vec{v}_{2,n} & \dots & \vec{v}_{n,n} & \vec{v}_{n+1,n} \end{array} \right] = \left[\begin{array}{ccccc|c} b_1 & b_2 & b_3 & \dots & b_n & b_{n+1} \\ b_2 & b_3 & b_4 & \dots & b_{n+1} & b_{n+2} \\ b_3 & b_4 & b_5 & \dots & b_{n+2} & b_{n+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_n & b_{n+1} & b_{n+2} & \dots & b_{2n-1} & b_{2n} \end{array} \right]$$

If the set of vectors, $\{\vec{v}_{1,n}, \vec{v}_{2,n}, \dots, \vec{v}_{n,n}\}$ is a linearly dependent set, then some of the $\alpha_{k,n}$ will be free variables. In this case, we want to set $\alpha_{k,n} = 0$ for the lowest possible values of k . Thus, we set $\alpha_{0,n} = \alpha_{1,n} = \dots = \alpha_{j,n} = 0$, for the largest value of j possible.

Once we have found the α_k , we create the companion matrix:

$$A_n = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 1 & & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ \alpha_{0,n} & \alpha_{1,n} & \alpha_{2,n} & \alpha_{3,n} & \dots & \alpha_{n-1,n} \end{bmatrix}$$

Note that we need to know b_{2n} in order to create the matrix A_n .

We see that the matrix A_n has the property that $A_n \vec{v}_k = \vec{v}_{k+1}$ for $1 \leq k \leq n$, and thus has the correct growth rate of \mathcal{G} for the first $2n$ terms of $\{b_k\}$.

The companion matrix, A_n , has characteristic polynomial:

$$p_{A_n}(x) = x^n - \alpha_{n-1}x^{n-1} - \alpha_{n-2}x^{n-2} - \dots - \alpha_1x - \alpha_0$$

From (8), we know that the α_k satisfy:

$$\alpha_0 b_k + \alpha_1 b_{k+1} + \alpha_2 b_{k+2} + \cdots + \alpha_{k-1} b_{2k-1} = b_{2k} \quad 1 \leq k \leq n$$

Alternatively, we write:

$$b_{2k} - \alpha_{k-1} b_{2k-1} - \alpha_{k-2} b_{2k-2} - \cdots - \alpha_2 b_{k+2} - \alpha_1 b_{k+1} - \alpha_0 b_k = 0 \quad 1 \leq k \leq n$$

Which we then write as:

$$\left(b_{\frac{1}{2k}}\right)^{2k} - \alpha_{k-1} \left(b_{\frac{1}{2k-1}}\right)^{2k-1} - \cdots - \alpha_1 \left(b_{\frac{1}{k+1}}\right)^{k+1} - \alpha_0 \left(b_{\frac{1}{k}}\right)^k = 0 \quad (9)$$

We know that for large values of k , $b_k^{1/k}$ is close to the correct entropy. Thus, each value in parentheses above is near the correct entropy for large enough k . That is to say that the entropy, 2^h , is *nearly* a solution of the characteristic polynomial of A_n when n is large.

We state this formally as:

Theorem 9.1 *Let \mathcal{G} be a one-dimensional shift of finite type, with alphabet \mathcal{A} and entropy h . Construct matrices A_n as above. Then there exists an N such that for $n \geq N$, $\rho(A_n) = 2^h$, where $\rho(A_n)$ denotes the spectral radius of A_n .*

Proof: Since \mathcal{G} is a shift of finite type, it has an associated transition matrix, $T_{\mathcal{G}}$. For purposes of this proof, we can let $N = |\mathcal{A}|$.

Let $p_{T_{\mathcal{G}}}(x) = x^N + a_{N-1}x^{N-1} + \cdots + a_1x + a_0$ denote the characteristic polynomial of $T_{\mathcal{G}}$.

Since $T_{\mathcal{G}}$ satisfies its characteristic polynomial, we know that:

$$p_{T_{\mathcal{G}}}(T)\vec{\mathbf{1}}_N = [T^N + a_{N-1}T^{N-1} + \cdots + a_1T + a_0I]\vec{\mathbf{1}}_N = \vec{\mathbf{0}} \quad (10)$$

$$\implies \| [T^N + a_{N-1}T^{N-1} + \cdots + a_1T + a_0I]\vec{\mathbf{1}}_N \|_1 = 0$$

Since we are only dealing with positive numbers, the 1-norm is linear:

$$\implies \|T^N\vec{\mathbf{1}}_N\|_1 + a_{N-1}\|T^{N-1}\vec{\mathbf{1}}_N\|_1 + \cdots + a_1\|T\vec{\mathbf{1}}_N\|_1 + \|a_0I\vec{\mathbf{1}}_N\|_1 = 0$$

$$\implies b_{N+1} + a_{N-1}b_N + \cdots + a_1b_2 + a_0b_1 = 0$$

Thus, by setting $\alpha_{i,N} = -a_i$, we satisfy the first equation in (8).

Furthermore, by multiplying equation (10) by T^k for $k \geq 0$, and following the same course, we satisfy every equation in (8) for $n = |\mathcal{A}|$. Note that we have actually solved infinitely many such equations, and that $A_{|\mathcal{A}|}$ will generate the entire sequence $\{b_n\}$ correctly.

The matrix $A_{|\mathcal{A}|}$ also has the same characteristic polynomial as $T_{\mathcal{G}}$, and thus

$$\rho(A_{|\mathcal{A}|}) = 2^h.$$

■

Underlying all of this is the fact that $\frac{1}{n} \log |\mathcal{B}_n(\mathcal{G})|$ is finite, and converges to 2^h . For a \mathbb{Z}^2 -action, we know that $\frac{1}{n} \log \lambda_n$ also converges to 2^h , where the λ_n are the entropies of the n -wide restrictions of the \mathbb{Z}^2 -action.

The natural question now is: "to what extent can the λ_n be expressed as a recursion?"

In the 2-dimensional setting, using the λ_n , we no longer have a finite alphabet, nor do we have a transfer matrix to exploit. However, we do know that the sequence of λ_n behave as if they were generated by an iterated linear operator in the sense that $\frac{1}{n} \log \lambda_n \rightarrow h$.

In the 2-dimensional setting, we do not expect that $\rho(A_n)$ will equal 2^h for any n . That is to say that we don't expect to have a finite-dimensional recursion which describes the sequence λ_n perfectly. However, the theorem tells us that the sequence is (in a sense) *nearly* a recursive sequence.

Having a numerical estimate for the entropy of a \mathbb{Z}^2 -action is nothing new. However, this algorithm is of particular interest because of the rate at which $\rho(A_n)$ converges to the correct entropy for the system.

We include MATLAB programs that use the substitution algorithm described in chapter 3 (**genalg_nz.m**), and the numerical approximation ($\rho(A_n)$) algorithm described here (**num_approx.m**) in appendix [A]. Entropy approximations were generated using these programs, and their errors were determined by using the hard square entropy approximation given by Baxter in [1], $2^h = 1.5030480824753322643220663294755536893857810 \dots$, which is known to be accurate to 43 decimal places. The MATLAB programs generated the following data, for the sequence $\{\lambda_i\}_{i=1}^n$

n	$\rho(A_n)$	Baxter's value $-\rho(A_n)$
2	1.49206603764754	$1.09820448277955 \times 10^{-2}$
4	1.50304256195751	$5.52051782665153 \times 10^{-6}$
6	1.50304806840618	$1.40691476246957 \times 10^{-8}$
8	1.50304808251233	$-3.69941854927447 \times 10^{-11}$
10	1.50304808247359	$1.74327219326642 \times 10^{-12}$
12	1.50304808247533	$5.99520433297585 \times 10^{-15}$

Since MATLAB is accurate to only 15 decimal places, the round-off error actually claims that when $n = 12$ that there is a recursion, and that the entropy estimate is equal to the correct entropy for the system. This is, of course, not true, and is due to the limitations of the computer used and in the MATLAB programs used to compute the eigenvalues of the V_n matrices and to row-reduce the Hankel matrices.

Also, we have strong reasons to believe that the $\rho(A_n)$ approximations should converge monotonically increasing to the correct entropy (that is, we believe that for each n , $\rho(A_n) \leq \rho(A_{n+1}) \leq H$). That being said, we believe that the $\rho(A_n)$ entropy estimate for $n = 8$ to be in error, as it represents an estimate that is actually *above* Baxter's entropy estimate.

However, it is worth noting that due to the triviality of this computation, these estimates were generated very quickly (under one half of a second).

For the 3-coloring model (which is known to be equivalent to the *six-vertex* or *square ice* model), we know the exact entropy to be $H = (4/3)^{3/2}$ (as computed for the six-vertex model in [14]). The same MATLAB programs generated the following estimates:

n	$\rho(A_n)$	$(4/3)^{3/2} - \rho(A_n)$
2	1.5	$3.96007178390021 \times 10^{-2}$
4	1.53319383707296	$6.40688076604068 \times 10^{-3}$
6	1.53773728723612	$1.86343060288174 \times 10^{-3}$
8	1.53887120431789	$7.29513521113612 \times 10^{-4}$
10	1.53925826509907	$3.42452739934718 \times 10^{-4}$

We know that the square ice model has long-term, non-trivial dependencies given certain $n \times n$ configurations. Thus, we see the $\rho(A_n)$ approximations converging more slowly than the Hard Square model (whose long term, non-trivial dependencies are very small, due to the "free square", \square).

The Domino system also exhibits long-term dependencies (the Domino system is known to be a factor of the six-vertex model). The entropy for the Domino system is computed in [5] to be approximately $2^h = 1.33851515197606\dots$, which we believe accurate to 12 decimal places. The MATLAB program approximations for the Domino system are:

n	$\rho(A_n)$	$2^h - \rho(A_n)$
2	1.33141483151058	0.00710032046548248
4	1.36472418661596	-0.0262090346398953
6	1.33745171080222	0.00106344117384238
8	1.51675433232394	-0.178239180347872

In this case, we see the approximations are quite poor, and do not even seem to be converging to the correct entropy. Furthermore, we also see that some of the approximations are above the correct entropy. Thus, we believe that these poor approximations are due to round-off errors.

Bibliography

10 Bibliography**References**

- [1] R. J. Baxter: *Planar Lattice Gases with Nearest-Neighbor Exclusion*. Annals of Combinatorics **3** (1999) 191-203.
- [2] R. J. Baxter, I. G. Enting, and S. K. Tsang: *Hard Square Lattice Gas*. Journal of Statistical Physics **22** (1980) 465-489.
- [3] P. Billingsley: *Ergodic Theory and Information* Wiley (1965)
- [4] R. Burton, J. Steif: *Non-Uniqueness of Measures of Maximal Entropy for Subshifts of Finite Type*. Ergodic Theory and Dynamical Systems **14** (1994) 213-235
- [5] R. Burton, R. Pemantle: *Local Characteristics, Entropy and Limit Theorems for Spanning Trees and Domino Tilings via Transfer-Impedances*. Annals of Probability **21** (1993) 1329-1371.
- [6] R. Burton, K. Dajani, and D. Meester: *Entropy for Random Group Actions*. Ergodic Theory and Dynamical Systems **18** (1998) 109-124
- [7] R. Burton, J. Steif: *New Results on Measures of Maximal Entropy*. Israel Journal of Mathematics **89** (1995) 275-300
- [8] N. J. Calkin and H. S. Wilf: *The number of independent sets in a grid graph*. SIAM Journal of Discrete Mathematics **11** (1998) 54-60.
- [9] K. Engel: *On the Fibonacci Number of an $M \times N$ Lattice*. The Fibonacci Quarterly **28** (1990) 72-78.
- [10] Friedland, Shmuel : *Computation of entropy in statistical mechanics and information theory*, Congr. Numer. 168 (2004), 207–213.

- [11] Hoffman, Chris: *Private conversations*
- [12] J. Kemeny, J.L Snell: *Finite Markov Chains*. D. Van Nostrand Company, Inc. (1960)
- [13] R. Kindermann, J. L. Snell: *Markov Random Fields and their Applications*. Contemporary Mathematics (American Mathematical Society) (1980).
- [14] E.H. Lieb: *Residual Entropy of Square Ice*. Physical Review Vol. 162, 1 (1967) 162-172
- [15] D. Lind, B. Marcus: *An Introduction to Symbolics and Coding*. Cambridge University Press (1995).
- [16] D. Lind: *Private conversations*
- [17] N.G. Markley and M.E. Paul, *Maximal measures and entropy for \mathbb{Z}^{ν} subshifts of finite type* (1979)
- [18] W. Rudin: *Principles of Mathematical Analysis, Third Edition*. McGraw-Hill (1976)
- [19] Paul Shields: *Private conversations*
- [20] Solomyak, Boris: *Private conversations*

A Matlab Code

This appendix shows the Matlab code for 4 of the programs used in this paper:

- `genalg.m` , which is the substitution algorithm outlines in section 3
- `genalg_nz.m` , which generates the same substitutions without the zero rows and columns
- `genalg2.m` , which outputs the Perron eigenvalues of the matrices generated by the substitution algorithm
- `num_approx.m` , which computes the numerical approximatino algorithm outlined in section 9

Note that some lines have been wrapped to fit on the page.

```

%%%%%%%% genalg:
% This is the general matrix substitution algorithm.
% The "seed" or input matrix, V2 (H2), should be (a^2)x(a^2) where
a is the
% alphabet size. k is the desired number of iterations. The output of
genalg.m
% will be the matrix Vk (Hk). (This version uses sparse matrix
notation.)
%
% Usage:   genalg(V2,k)

function F=genalg(V2,k)

% A will be the alphabet size:
A=sqrt(max(size(V2)));

% Special function in case k=1, in which we compute the
% V1 matrix.

if k==1

    % Seed matrix for V1:
    Z=zeros(A);

    for r=1:A
        for c=1:A
            if V2( (r-1)*A+1:(r*A) , (c-1)*A+1:(c*A) )==zeros(A)
                Z(r,c)=0;
            else
                Z(r,c)=1;
            end
        end
    end

```

```

        end
    end
    F=Z;
    break
end

% Special function in case k=2, in which we return the original
matrix:

if k==2
    F=V2;
    break
end

% The substitution algorithm for k > 1 ; V_n=T ; V_{n+1}=T2 ;
then repeat:
T=V2;

% start the iterations. Iterate by creating a T2 which which all
submatrices
% are T, then putting in the appropriate zeroes:
% First, find all of the zeros in V2 so we know where to put
zero blocks:
[ZR,ZC]=find(V2==0);
NZB=max(size(ZR));

for iter=3:k
    T2=zeros(A^iter);

%   Make a bunch of copies of T:

    for SUBROW=1:A
        for SUBCOL=1:A
            T2( (SUBROW-1)*A^(iter-1)+1:(SUBROW)*
A^(iter-1) , (SUBCOL-1)*A^(iter-1)+1:(SUBCOL)*A^(iter-1) ) = T;
            end
        end

%   Put zero blocks in the correct places:
        for i=1:NZB
            T2( (ZR(i)-1)*A^(iter-2)+1:(ZR(i))*A^(iter-2) , (ZC(i)-1)*
A^(iter-2)+1:(ZC(i))*A^(iter-2) ) = zeros(A^(iter-2));
            end

%   T2 (which is V_{n+1}) is now finished.

```



```
    T=T2;  
end  
F=T;
```

```

%%%%% genalg_nz:
% This is the general matrix substitution algorithm without
zero rows and columns.
% The "seed" or input matrix, V2 (H2), should be (a^2)x(a^2)
where a is the
% alphabet size. k is the desired number of iterations. The
output of genalg_nz.m
% will be the matrix Vk (Hk). The matrix, V2, must be a binary
matrix.
%
% Usage:   genalg_nz(V2,k)
%

function Vk=genalg_nz(V2,k)

% A will be the alphabet size:
A=sqrt(max(size(V2)));

% When we eliminate the zero rows from the start, the
substitution
% matrices are no longer square. We generate a matrix, PM,
which will
% keep track of how to correctly partition the Vn matrices.
% The correct partition will be PM^(n-1)*x , where x is a vector
% with a one in every entry. This partition will be saved as the
% vector, PV. We will also need the partition for the matrix to
% be created, which we will call PV2.

x=ones(A,1);
PM=genalg(vh(V2),1);

% Now we will create two matrices, ROWSUB, and COLSUB
to denumerate the
% substitutions.
% In order to make these matrices, we first need to create an
intermediate
% matrix, E, which enumerates the rows and columns.
E=[ ];

for m=1:A
    E=[E; m*ones(1,A^2) ];
end

E=kron( ones(A,1) , E );

ROWSUB=E;

```

```

COLSUB=E';

% ROWSUB and COLSUB hold all of the information needed
% for the
% substitution algorithm.

% We need to strip the zero rows and columns from V2,
% ROWSUB, and COLSUB

VN=V2;

for n=A^2:-1:1
    if sum(VN(:,n))==0
        VN(:,n)=[];
        ROWSUB(:,n)=[];
        COLSUB(:,n)=[];
    end
end

for n=A^2:-1:1
    if sum(VN(n,:))==0
        VN(n,:)=[];
        ROWSUB(n,:)=[];
        COLSUB(n,:)=[];
    end
end

VS2=VN;

% VN should be a square matrix, whose size is the "real"
% alphabet
% size. We call RA the real alphabet size:

RA=max(size(VN));

PV=PM*x;
PV2=PM*PV;

for r=1:k-2

% We now create an Ax2 block-counting matrix, BC,
% which holds the
% values of the places where the partitions take place.
% For instance,
% if PV=[ 3 ; 5 ; 7 ] , then BC=[ 1 3 ; 4 8 ; 9 15 ], which are

```

```

    the
% indices for the substituted rows and columns.
    (i.e. 1:3 , 4:8 , 9:15)
% Similarly, we define BC2 for the next matrix.

BC=zeros(A,2);
for n=1:A
    BC(n,1)= sum(PV(1:n)) - PV(n) + 1;
    BC(n,2)= sum(PV(1:n));
end

BC2=zeros(A,2);
for n=1:A
    BC2(n,1)= sum(PV2(1:n)) - PV2(n) + 1;
    BC2(n,2)= sum(PV2(1:n));
end

% Now to create the VNP (V_{n+1}) matrix block by
    block:

VNP=[];

for RB=1:RA

    ROWBLOCK=[];

    for CB=1:RA

        ROWBLOCK=[ ROWBLOCK [VS2(RB,CB)*VN( BC(
ROWSUB(RB,CB),1):BC(ROWSUB(RB,CB),2) , BC(
COLSUB(RB,CB),1):BC(COLSUB(RB,CB),2) )] ];

    end

    VNP=[VNP;ROWBLOCK];

end

% Just so we can see of which eigenvalue was just
    computed:
sprintf('%d',r+2)

%%%
%
% VNP is now created. We set our new values for the

```

```
        next iteration
    %
    %%
    VN=VNP;
    PV=PV2;
    PV2=PM*PV;

    end

    Vk=VNP;
```

```

%%%%% genalg2
% Use the genalg algorithm to output a column vector
% whose entries are the Perron Eigenvalues of each
% iteration of the substitution algorithm.

function M=genalg2(V2,k)

% Seed the column vector to hold the eigenvalues
E=[ max(eigs3(genalg(V2,1))) ; max(eigs3(V2)) ];

% A will be the alphabet size:
A=sqrt(max(size(V2)));

% The substitution algorithm for k > 1 ; V_n=T ; V_{n+1}=T2 ;
then repeat:
T=V2;

% start the iterations. Iterate by creating a T2 which which all
submatrices
% are T, then putting in the appropriate zeroes:
% First, find all of the zeros in V2 so we know where to put
zero blocks:
[ZR,ZC]=find(V2==0);
NZB=max(size(ZR));

for iter=3:k

    T2=zeros(A^iter);

%   Make a bunch of copies of T:
    for SUBROW=1:A
        for SUBCOL=1:A
            T2( (SUBROW-1)*A^(iter-1)+1:(SUBROW)*
                A^(iter-1) , (SUBCOL-1)*A^(iter-1)+1:(SUBCOL)*
                A^(iter-1) ) = T;
        end
    end

%   Put zero blocks in the correct places:
    for i=1:NZB
        T2( (ZR(i)-1)*A^(iter-2)+1:(ZR(i))*A^(iter-2) , (ZC(i)-1)*
            A^(iter-2)+1:(ZC(i))*A^(iter-2) ) = zeros(A^(iter-2));
    end

%   T2 (which is V_{n+1}) is now finished.
    T=T2;

```

```
E=[ E ; max(eigs3(T)) ];  
end  
M=E;
```

```

%%%% num_approx
% Numerical approximation for entropy using the
% companion matrix method.
%
%
% Usage: [P,H]=num_approx(V2,n) where n is an even
integer representing
% largest eigenvalue computed and V2 is the usual input
matrix.
%
% i.e.: num_approx(V2,8) generates 8 eigenvalues, giving the
% Fourth approximation.
%
% P will be a column vector of coefficients of the characteristic
% polynomial of the n/2 partner matrix (Note the first entry will
% always be one, since the polynomial is monic), and H is the
% largest root of this polynomial.

function [P,H]=num_approx(V2,m)

% For convenience in programming:

n=m/2;

% Generate the column vector of eigenvalues

V=genalg2(V2,m);

% Create an empty matrix, R, to hold the eigenvalues
% for the rref to generate the companion matrix:

R=zeros(n,n+1);

for c=1:n+1

    R(1:n,c)=V(c:c+n-1);

end

A2=rref(R);

% The order of the entries needs to be reversed to
% be able to use matlab's root function properly.

Y=zeros(n,1);

```



```
for e=0:n-1

    Y(e+1)=-A2(n-e,n+1);

end

% A will be the final column vector generated by row
reduction
% of the matrix R. That is, A2 transpose will be the entries
of the bottom
% row of the companion matrix.

P=[1;Y];

% The coefficients are in the correct order. We need to add
% one to the degree to make a monic polynomial of degree
n+1
% then find the largest root.

% H will be the approximation of the entropy given this
polynomial

H=max(abs(roots([1 ; Y]))) ;
```