Summer 8-7-2020

A Theory of Preimage Complexity: Data-structures, Complexity Measures and Applications to Endofunctions and Associated Digraphs

Bradford M. Fournier-Eaton
University of New Orleans, bradfordfournier@gmail.com

A Theory of Preimage Complexity:
Data-structures, Complexity Measures and Applications to Endofunctions and Associated Digraphs

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements of the degree of

Doctor of Philosophy
in
Engineering and Applied Science
Mathematics

by

Bradford M. Fournier-Eaton

B.S. University of New Orleans, 2013
M.S. University of New Orleans, 2016

August, 2020

For Scott

Acknowledgments

I would like to express my sincere gratitude to my committee for their patience, kindness and encouragement throughout this process. To my advisor and mentor Dr. Kenneth Holladay -- a man of great generosity, warmth, and infectious enthusiasm: I would like to express my most heartfelt thanks for his insight, creativity, and trust -- without which this dissertation would not have been possible. Additional thanks to my dear friend and colleague Aram Bingham whose companionship, conversation, support and encouragement has for many years been a source of strength and whose character and kindness is a model of friendship to which I can only hope to aspire. Finally to my husband Scott: your kindness of spirit, ever-present love and unflinching support have made even the most difficult moments in this journey a true gift -- together we can indeed do anything. Additional thanks to Dr. Li, Dr. Golz, Dr. Easterlin, and Dr. Ahmed.

Foreword

Only briefly does the predictable and the understood hold sway over our human thirst for the captivating and the beautiful: the kaleidoscope of nature, complexity of peak and valley, textures of schist and chitin, the care-free path of fjord and tributary, and swells of foam and grain. We marvel at Stravinsky, Pynchon and Pollak. We exclaim the unbelievability, the near impossibility. We imagine a young genius working without effort, or living a life unceasing in toil.The above examples speak to the difficulty of tracing causality in complex systems and also invoke the notion of beauty. It is the seemingly untraceable course to beauty which we find so compelling: Stone Forrest in Shilin China, Colombia's Cano Christales River, Paro Satang Monastery of Bhutan, and the Fairy Pools in The Isle of Skye. We are most dumfounded when standing silently, stupefied and still, before beauty while wondering at history and cause. These wonders have causes, histories, and futures of their own. To know the partial histories of these places and forces diminishes neither their beauty nor our awe. The forces and phenomena at work here share a complex beauty and inspire our imagination to ponder their origins. "Origin" is the motivating philosophical impulse of this thesis as we embark on a study of discrete causality and complexity. Complexity and richness of structure need not evolve from complex rules; simple rules along with simple initial conditions can give rise to vastly different complexities: from the simple to as rich as any found in nature.[1]

Contents

List of Figures……………………………………………………………………………………vi

Abstract……………………………………………………………………………………………vii

Introduction…………………………………………………………………………………………1

Preimage Data Structures……………..……………..………………………………………………3

Results on Sigma Matrices…………………………………………………………………………12

Graphs and Sigma Matrices………………………………………………………………………17

Preimage Complexity Functions…………………………………………………………………22

Sequence of Induced Partitions and A Novel Sequence…………………………………………32

Directions For Future Research…………………………………………………………………35

References…………………………………………………………………………………………38

Appendices…………………………………………………………………………………………39

Vita…………………………………………………………………………………………………42

List of Figures

Figure 4.1: A cyclic digraph of order 7 18

Figure 4.2: A functional tree of order 8..20

Figure 4.3 A directed acyclic graph...20

Figure 5.1 The signature of 4×4 sigma matrix..25

Abstract

This dissertation develops a new theory of finite function complexity. This novel approach is based on the structure of preimage sets generated under repeat application of the inverse. We encode this information in our primary data-structure, an square matrix called the sigma matrix. This matrix allows us to easily encode information about the functional digraph and cycle structure of the associated endofunction. Additionally, the sigma matrix is of interest in its own right. The columns of sigma matrices are integer partitions of the domain size n, the size of the domain is always an eigenvalue of the sigma matrix, and calculation of the sigma matrix is highly efficient -- requiring no direct calculation of inverses. The problem of finding the number of unique sigma matrices on X^X as a function of n, the size of X, gives rise to a novel integer sequence $\Sigma(n)$. We use the sigma matrix and a natural ordering on these matrices as part of a flexible and informative definition of preimage complexity. We give several examples of preimage complexity measures and examine the partition of all endofunctions induced by these measures. Finally we show how the integer sequence $\Sigma(n)$ corresponds to the number of complexity classes induced by the discrete preimage complexity function.

Keywords: Preimage Structures, Endofunctions, Sigma Matrices, Integer Partitions, Integer Sequence

1. Introduction

In this dissertation we develop the theory of preimage complexity of endofunctions on a finite domain. We roughly divide this work into two parts: the first develops the data-structures required in our analysis of endofunction complexity and the second develops a flexible measure of a preimage complexity measure. The primary data-structure developed, the sigma matrix with entries given by $\Sigma[i, j] = |f^{-j}(x_i)|$, is designed to efficiently capture the preimage structure of the associated functions and is itself of mathematical interest. The columns of the sigma matrix are integer partitions of $n = \left|dom(f)\right|$, the matrix is efficiently calculated, and the number of such matrices as a function of n is a novel integer sequence. Additionally, the sigma matrix may be used to derive useful information about the digraph associated with a given endofunction. It is upon this structure that we develop a flexible measure of preimage complexity. Our preimage complexity functions take each function $f : X \to X$ to some $s \in S$, a countable set, by way of the sigma matrix. Thus at its most basic, a preimage complexity function is just a function $H : X^X \to \Sigma_{X^X} \to S$ where Σ_{X^X} is the set of all sigma matrices for the functions X^X and S the set in which the complexity value of f is assigned.

We use five benchmarks by which to judge our progress: two of these pertaining to the data-structures and three pertaining to our desired complexity measure. The data structures developed should capture the preimage structure of the function and encode useful information about the associated function and digraph (B1). Additionally the data structures should be easy to compute for large domains (B2). For our complexity measures we required that they be based on the preimage data structures (B3), be flexible and informative (B4), and respect the natural dichotomy between the preimage structure of the bijection and constant (B5).

The first benchmark demands that our structures readily correspond to the reality of the function's inverse structure. By simply examining the chosen structures, we should gain insight into the preimage structure of f. The second benchmark requires that the structures be computable in a simple way. Our definition of complexity will be general enough to flexibly partition the set of all endofunctions depending on the needs of the scientist using the measure: a highly discriminating measure may permit a fine partitioning of the functions into equivalence classes only when the functions have equal preimage structures whereas coarser measures may look at qualities beyond just simple equality. Lastly, we must be able to learn something useful about the function and its structure, often encoded as a digraph.

Looking Forward. In Section 2 we define the data structures which will be useful in talking about our functions in the language of preimages: preimage sets as well as image and preimage vectors. Additionally there are composite structures built from the above vectors: the forward image array, preimage array and the sigma matrix. These structures will form the basis from which we organize information about our functions and serve as efficient objects upon which to operate and derive a function's associated inverse properties. In keeping with our goals of the last section, the structures in Section 2 are designed to obey certain niceness properties making them useful, efficient, and applicable -- as well as being interesting objects in their own right. At the end of Section 2 we fully develop our primary data structure: an array of set sizes which will play a major role in how we structure and derive our results. Section 3 proves a number of interesting results about the structures developed in section two. In particular, we prove results about the primary data structure of this paper, the sigma matrix. In section 4 we show the applicability of our approach: extracting from our data-structures information about the associated digraphs. Section 5 develops a notion of preimage complexity: desired properties are outlined and justified, the definition is given, and several examples of preimage complexity functions are given. Section 6 examines the partitions of X^X induced by our preimage complexity functions and the sequence of such partitions as a function of $\left|\operatorname{dom}(f)\right|$. We exhibit a novel sequence $\Sigma(n)$ corresponding to the number of equivalence classes of complexity induced by our primary data structure as a function of n. Lastly, in Section 7 several avenues of further exploration and possible lines of inquiry are suggested.

2. Preimage Data-Structures

2.1 Notation and Conventions. In this section we lay out our notational conventions for dealing with preimage elements and structures. Our primary structures are lists and matrices - the elements of which are often resultant from several applications of the inverse of the function. We start by laying out conventions for the indexing of domains, define the j^{th} inverse of $x \in \text{dom}(f)$ is and examine the creation of lists of sets wherein the list ranges over domain elements under a fixed number of applications of the inverse. Likewise, we examine the case where the list ranges over increasing applications of f^{-1} for some domain element x. Forth and finally, we combine the above and examine a matrix, or list of lists, where for each i^{th} domain element x_i we find its $j-$ back inverse for all $1 \leq i, j \leq n$.

Unless otherwise noted, function domains will be in capital letters, usually X, and have size n. The elements of such a domain will be indexed lower-case versions of the letter used for the domain. The typical index for a domain element will, as above, be i and j will index the inverse depth where $1 \leq i, j \leq n$. I.e., given $f: X \to X$ we have $|X| = \left|dom(f)\right| = n$ where $X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$. Thus, recognize $f^{-j}(x_i)$ as is the preimage of $x_i \in X$ under j applications of the inverse of $f: X \to X$ and $|X| = n$.

For example, consider the immediate preimage set to an element y in the domain of a function, i.e., $f^{-1}(y) = \{x : f^1(x) = y\}$. Extending the inverse depth to provide a more complete genealogy of y under f, we may consider $f^{-j}(y) = \{ x : f^j(x) = y\}$ where $1 \leq j \leq n$. Likewise suppose we wish to find the j^{th} inverse across several domain elements. Suppose $A = \{a, b, c\}$ to find $f^{-j}(A)$ simply calculate $f^{-1}(A) = f^{-1}(a) \cup f^{-1}(b) \cup f^{-1}(c)$. Thus we use the convention that

$$f^{-j}(A) := \bigcup_{x \in A} f^{-j}(x). \tag{1}$$

Often we will create a list of preimage sets for a particular $x \in X$ as we apply the inverse multiple times. To keep such an accounting, we create a list of inverse sets of x at depths ranging from 1 to n, using brackets to indicate the range of values over which the bracketed index ranges. For example, if we wish populate a list with the inverses $f^{-j}(x)$ at depths from the first to n^{th} inverse, we create the n element list of such elements using brackets to indicate the values over which j should range,

$$f^{-[1,n]}(x) := \left[f^{-1}(x),\ f^{-2}(x),\ \ldots,\ f^{-n}(x)\right]. \tag{2}$$

To construct a list which iterates over all n domain elements while holding the inverse depth at a constant j, we place brackets around the set of elements which f^{-j} should take as arguments. Typically we do this for the set of all n domain elements $X = \bigcup_{i=1}^{n} x_i$ as in

$$f^{-j}[X] := \left[f^{-j}(x_1),\ f^{-j}(x_2),\ \ldots,\ f^{-j}(x_n)\right]. \tag{3}$$

In summary, brackets enclose the range of values which varies with the indexing of the list elements. For example, in $f^{-[1,j]}(x)$, it is the inverse depth which varies: ranging over 1 to j for the single element x. However in $f^{-j}[X]$ the inverse depth j is fixed and we consider this j^{th} inverse as the domain elements range from x_1 to x_n.

Example 2.1. Let $f = \{(a,\ b),\ (b,\ a),\ (c,\ a),\ (d,\ b)\}$.

If we wish to construct the list of the three-back inverses for each domain element in $X = \{a,\ b,\ c,\ d\}$ we write the following

$$f^{-3}[X] = \left[f^{-3}(a),\ f^{-3}(b),\ f^{-3}(c),\ f^{-3}(d)\right].$$

To examine the set of all such elements at an inverse depth of three, we find

$$f^{-3}(X) = \bigcup\left(f^{-3}(a),\ f^{-3}(b),\ f^{-3}(c)\right) = \{a,\ b,\ c,\ d\}.$$

Finally, for a fixed element b, we can list the preimage sets at depths 1 to n. For the element b,

$$f^{-[1,n]}(b) = \left[f^{-1}(b), f^{-2}(b), f^{-3}(b)\right] = [\ \{a,\ d\},\ \{b,\ c\},\ \{a,\ d\}\ ].$$

We want a list of lists capture the complete preimage picture of a function and summarize the above in the following definition.

Definition 2.2. The matrix of preimage sets for each domain element x_i and each inverse depth $1 \leq j \leq n$ will be called the *preimage matrix* F^{-}. The entry in the i^{th} row and j^{th} column is given by $\mathrm{F}^{-}[i,\ j] := f^{-j}(x_i)$. This may be seen as a matrix of rows with i^{th} row corresponding to the list $f^{-[1,n]}(x_i)$.

Similarly this can be viewed as a column matrix with the j^{th} column corresponding the list $f^{-j}[X]$. We use the notation $f^{-[1,n]}[X]$ to indicate such a matrix which iterates over both domain elements $[X]$ and inverse depth $[1, 2, \ldots, n]$.

$$F^{-} := f^{-[1,n]}[X] = \begin{pmatrix} f^{-1}(x_1) & f^{-2}(x_1) & \cdots & f^{-n}(x_1) \\ f^{-1}(x_2) & f^{-2}(x_2) & \cdots & f^{-n}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f^{-1}(x_n) & f^{-2}(x_n) & \cdots & f^{-n}(x_n) \end{pmatrix}. \tag{4}$$

Example 2.3 Let $f\{(a, b), (b, a), (c, a), (d, b)\}$

The preimage matrix F^{-} is the list of lists $f^{-[1,n]}[X]$. The first and second column of this matrix are given by the lists

$$f^{-1}[X] = \left[f^{-1}(a), f^{-1}(b), f^{-1}(c), f^{-1}(d)\right] = [\{b, c\}, \{a, d\}, \{\}, \{\}]$$

and

$$f^{-2}[X] = \left[f^{-1}(f^{-1}(a)), f^{-1}(f^{-1}(b)), f^{-1}(f^{-1}(c))\right]$$

The first element of the second column, $F^{-}[1, 2] = F^{-}_{(1,2)}$ is $f^{-2}(x_1 = a)$ which we calculate as

$$f^{-2}(a) = f^{-1}(f^{-1}(a)) = f^{-1}(b, c) f^{-1}(b) \cup f^{-1}(c) = \{a, d\}.$$

Continuing for the each $1 \le i, j \le 4$ we populate the preimage matrix as

$$F^{-} = \begin{pmatrix} \{b, c\} & \{a, d\} & \{b, c\} & \{a, d\} \\ \{a, d\} & \{b, c\} & \{a, d\} & \{b, c\} \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}.$$

We now briefly preview a motivation for the next sub-section. Suppose we are given two functions f and g. Assume the preimage matrices of f and g are

$$F^{-} = \begin{pmatrix} a & a & a \\ b & b & b \\ c & c & c \end{pmatrix} \text{ and } G^{-} = \begin{pmatrix} c & b & a \\ a & c & b \\ b & a & c \end{pmatrix}.$$

Notice that each of these preimage matrices are made of singleton entries. So, for each additional inverse step there is a unique elemental inverse. Functions like f and g above with unique inverse elements should be recognized as bijections. Thus in addition to the particular elements making up the inverse sets, the size of the inverse sets themselves gives useful information. Capitalizing on this feature, we develop a structure that captures the *sizes* of the elements of the preimage matrix to exploit the information contained therein:

the sigma matrix.

2.2 The Sigma Matrix. We now turn to developing our primary object for the analysis of preimage structures. Recall that the preimage matrix F^{-} is an $n \times n$ matrix of rows populated with set elements such that the row number i row corresponds to a unique domain element x_i and the column number j to the inverse depth f^{-j}. Thus, $F^{-}[i,\ j] = f^{-j}(x_i)$.

Definition 2.4. The *Sigma Matrix* Σ_f *of* f is an $n \times n$ matrix with entries equal to the size of the corresponding entries in the preimage matrix. I.e., $\Sigma_f[i,\ j] := |F^{-}[i,\ j]| = |f^{-j}(x_i)|$.

$$\Sigma_f := \begin{pmatrix} |f^{-1}(x_1)| & |f^{-2}(x_1)| & \cdots & |f^{-n}(x_1)| \\ |f^{-1}(x_2)| & |f^{-2}(x_2)| & \cdots & |f^{-n}(x_2)| \\ \vdots & \vdots & \vdots & \vdots \\ |f^{-1}(x_n)| & |f^{-2}(x_n)| & \cdots & |f^{-n}(x_n)| \end{pmatrix}$$

We define that two sigma matrices Σ_1 and Σ_2 are *equal* if and only if there exists a permutation of rows π so that the two are element-wise equal.

2.3 Computation of Sigma Matrices. Computing the sigma matrix Σ_f by a naieve element-wise determination and count of preimage elements $F^{-}[i,\ j] = f^{-j}(x_i)$ for all $1 \le i,\ j \le n$ is not without its costs. For example, creating the three-by-three sigma matrix of $f = \{(a,\ b),\ (b,\ a),\ (c,\ b)\}$ requires us to first create the $n^2 = 9$ sets to populate the preimage matrix F^{-}. For just the first row of the matrix, $F^{-}[x_1,\ *] = F^{-}[a,\ *]$, we need $f^{-j}(a)$ for each of $j = 1,\ 2,\ 3$.

$f^{-1}(a) = \{b\}$,

$f^{-2}(a) = f^{-1}(b) = \{a,\ c\}$, and

$f^{-3}(a) = f^{-1}(a) \bigcup f^{-1}(c) = \{b\} \bigcup \{\} = \{b\}$.

So, here we would produce $F^{-}[a,\ *] = [\{b\},\ \{a,\ c\},\ \{b\}]$ as the first row. Next, we need the sizes of the aforementioned list elements to populate the sigma matrix. Again we only calculate the first row corresponding to the element $x_1 = a$ giving $\Sigma_f[a,\ *] := [|\{b\}|,\ |\{a,\ c\}|,\ |\{b\}|] = [1,\ 2,\ 1]$. Since this only corresponds to $a \in X$ we would need to repeat the process for $x = b$, and $x = c$. To call this method cumbersome is generous. One of our goals is to define and compute our preimage structures in a simple way: the above approach will not suffice. To enable a far more efficient method for calculation of the sigma matrix, we define two matrices: the image matrix F^+ and the column accumulation matrix ColSum(M). In addition we define two lists which make up the rows of the respective matrices.

Definition 2.5 : The *image list of x under f*, is the list of images of x under repeat application of f from f^1 to f^n. We write $f^{[1,n]}(x) = [\, f(x), f^2(x), \ldots, \; f^{n-1}(x), \; f^n(x)\,]$.

Definition 2.6 : The **image matrix of f** is the matrix in which the rows are the image lists of f for each x. We write $F^+ := f^{[1,n]}[X] = [f^{[1,n]}(x_1), f^{[1,n]}(x_2), \; \ldots, \; f^{[1,n]}(x_n)]$. Thus an arbitrary element of the image matrix is given by $F^+[i, j] = f^j(x_i)$.

Example 2.7. Let $f = \{(a, b), (b, c), (c, c)\}$. Find the image list of a under f and the image matrix of f.

The image list of a under f is the list with $j$thentry $f^j(a)$ by Definition 2.5 Here since we only have three domain elements, we are looking for the list $f^{[1,3]}(a)$ which is $[f(a), f^2(a), f^3(a)] = [b, c, c]$. By Definition 2.6, $F^+(a) = f^{[1,3]}(a)$ as calculated above. We need to find the remaining rows $F^+[b, *]$ and $F^+[c, *]$. These rows, along with the first are given by

$$F^+[a, *] = f^{[1,3]}(a) = \left[f(a), f^2(a), f^3(a)\right] = [b, c, c]$$

$$F^+[b, *] = f^{[1,3]}(b) = \left[f(b), f^2(b), f^3(b)\right] = [c, c, c]\,.$$

and

$$F^+[c, *] = f^{[1,3]}(c) = \left[f(c), f^2(c), f^3(c)\right] = [c, c, c]\,.$$

This forms the matrix:

$$F^+ = \begin{pmatrix} b & c & c \\ c & c & c \\ c & c & c \end{pmatrix}$$

Now we define a matrix which, in a given ith row, counts the number of x_i's in each column of some other matrix M. We can define the rows of such a matrix in the following definition.

Definition 2.8 : Let L be of length n and M be an $n \times n$ matrix. L is a **column accumulator of x over M** when the j^{th} entry $L[j]$ counts the occurrences of x in the jth column of M. We write ColSum($M : x$) and ColSum($M : x$) $[j]$ for the accumulator list of x over M and its $j$thentry respectively.

We now define a matrix built from the column accumulator lists of some matrix M ColSum$(M : x)$ as its rows.

Definition 2.9. A matrix M is a **column accumulation matrix of** M when the rows of M are the column accumulator lists of M. This matrix has entries $M[i, j] :=$ ColSum$(M : x_i)[j]$ and we write ColSum$(M : X)$ or just Colsum(M).

Example 2.10. Find the Column Accumulation Matrix of M $= \begin{pmatrix} n & m & n \\ l & n & m \\ l & l & m \end{pmatrix}$

1. First for each unique entry x of $M \in \{ * \}^{n \times n}$ we find the column accumulator list Colsum$(M : x)$. Recall that this is done element-wise by finding the number of x' s in in each column of M so that Colsum$(M : x)$ is the list [Colsum$(M : x)[1]$, Colsum$(M : x)[2]$, Colsum$(M : x)[3]$]. Doing this for $x = l$: there are two l's in column one, one l in column two and none in column three. Thus we have Colsum$\big(M : l\big)[1] = 2$, Colsum$\big(M : l\big)[2] = 1$, and Colsum$\big(M : l\big)[3] = 0$. These values populate the list Colsum$\big(M : l\big) = [2, 1, 0]$. Repeat this for elements m and n giving Colsum$\big(M : l\big) = [2, 1, 0]$, Colsum$(M : m) = [0, 1, 2]$, Colsum$(M : n) = [1, 1, 1]$.

2. We now find Colsum$(M) :=$ Colsum$(M : X)$. From the definitions this is the list $\big[$Colsum$\big(M : l\big)$, Colsum$(M : m)$, Colsum$(M : n)\big]$. Using the lists found in part 1 above, we form the matrix

$$\text{Colsum}(M) = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}.$$

Example 2.11. *Let* $g = \{(a, d), (b, d), (c, e), (d, f), (e, d), (f, e)\}$. We find another column accumulator matrix, here of an image matrix G^{+}.

$$G^{+} = g^{[1,n]}[X] = \begin{pmatrix} d & f & e & d & f & e \\ d & f & e & d & f & e \\ e & d & f & e & d & f \\ f & e & d & f & e & d \\ d & f & e & d & f & e \\ e & d & f & e & d & f \end{pmatrix}$$

We now find the $\mathrm{ColSum}(G^+) = [\mathrm{ColSum}(G^+ : a), \mathrm{ColSum}(G^+ : b), \ldots, \mathrm{ColSum}(G^+ : f)]$. To calculate, for example, the fifth row, $\mathrm{ColSum}(G^+ : x_5 = e)$, we count the number of m's in each column of G^+. This gives us the list $[2, 1, 3, 2, 1, 3]$. We proceed the same way for each $x \in \mathrm{dom}(g) = \{x_1 = a,\ x_2 = b,\ \ldots,\ x_{n=6} = f\}$. This results is the column accumulator matrix of G^+.

$$\mathrm{ColSum}(G^+) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 3 & 2 & 1 \\ 2 & 1 & 3 & 2 & 1 & 3 \\ 1 & 3 & 2 & 1 & 3 & 2 \end{pmatrix}$$

2.4 Computing the Sigma Matrix. In the last section we presented several widgets for which will be instrumental in computing the sigma matrix, our primary data structure of interest. Of particular interest will be the image matrix and the column accumulator matrix. Recall that the image matrix is simply the matrix with entries $F^+[i, j]$ corresponding to the image of x_i under j applications of f. The column accumulator matrix of M was a matrix $M_{\mathrm{cs}} = \mathrm{ColSum}(M)$ where the $(i, j)^{th}$ is given by the number of x_i's in column j of M.

$$M_{\mathrm{cs}}[i, j] = \mathrm{ColSum}(M : i)[j] = \mathrm{ColSum}(M)[i, j]$$

These two matrices, F^+ and $\mathrm{ColSum}(M)$ will be instrumental in an efficient calculation of the sigma matrix. Since sigma matrix was defined as the sizes of the preimage matrix entries: $\Sigma_f[i, j] = |f^{-j}(x_i)|$ we may ask what place an image matrix and a column accumulation matrix have in this calculation. Because nearly all of the work that follows will be using sigma matrices as its foundation, the efficient computation of these matrices is critical. We prove a rather surprising result that we first state as a claim.

Claim: The sigma matrix, a matrix of preimage set sizes, can be constructed without ever calculating a preimage set.

Theorem 2.12. *The sigma matrix Σ_f is the column accumulator matrix of F^+*

Proof: Call the image matrix F^+ and its column accumulator matrix C. Since C is the column accumulator matrix of F^+, the matrix element $C[i, j] = \mathrm{ColSum}(F : x_i)[j]$ counts the number of x_i in the j^{th} column of F. This is just a count of the x_i' s in $[f^j(x_1),\ f^j(x_2),\ \ldots,\ f^j(x_n)]$. For each occurrence of x_i in column j there is a corresponding element x so that $f^j(x) = x_i$. For example, if x_i occurs in position k of $f^j[X]$, then $x_k \in \{x : f^j(x) = x_i\}$. Note that since each occurrence of x_i corresponds to a unique row k, it corresponds to a unique domain element x_k and thus the number of x_i's in column j is exactly the same as the size of $\{x : f^j(x) = x_i\}$. I.e.,

$$C[i, j] = \text{ColSum}(F^+ : x_i)[j] = \left|\{x : f^j(x) = x_i\}\right| \tag{5}$$

Recall that the element $F^-[i, j]$ of the preimage matrix, is itself given by the set $f^{-j}(x_i) = \{x : f^j(x) = x_i\}$. Taking absolute values, we see that $C[i, j] = \text{ColSum}\left(F^+ : x_i\right)[j]$ is equal to the size of the preimage matrix element, $F^-[i, j]$. By definition of the sigma matrix, its (i, j) entry is the size of the $F^-[i, j]$ which gives us that

$$C[i, j] = \text{ColSum}(F^+ : x_i)[j] = \left|F^-[i, j]\right| = \left|f^{-j}(x_i)\right| = \Sigma_f(i, j). \tag{6}$$

$$\therefore$$

$$\text{ColSum}(F^+) = \Sigma_f \tag{7}$$

Q.E.D.

Theorem 2.12 above is crucial: it shows that we can circumvent the cumbersome element-wise creation of the sigma matrix from the preimage elements-- despite the fact that the sigma matrix element is given by $\Sigma[i, j] = |f^{-j}(x_i)|$. We simply need to calculate the $\text{ColSum}(F^+ : x)$ for each $x \in \text{dom}(f)$. This requires no calculation of any inverses as Σ_f is just the result of $\text{ColSum}(F^+)$, itself is simply a function of the image matrix: a matrix of singletons built from a simple procedure: repeat application of f. It should be noted ColSum is simply a function which takes as argument a matrix of singletons. The ColSum does not always generate a sigma matrix. The input to ColSum must be the image matrix for some $f : X \to X$. Thus (7) tells us that applying ColSum to an image matrix of f yields its sigma matrix. See Example 2.10 for an example of $\text{ColSum}(M)$ which does not result in a sigma matrix.

At this point we wish to distinguish between the statement that some matrix M is known *a priori* to be a sigma matrix, and that M is a sigma matrix derived from application of ColSum to an image matrix F^+ of some known function f. We will use the following nomenclature: we say that *M is a sigma matrix* if we know there exists at least one f so that $\text{ColSum}(F^+) = M$. In this case we typically will employ simply Σ or indicate that M "is a sigma matrix". However, we say that M is *the sigma matrix of f* if we have been given an f and know its sigma matrix is M having carried out a construction. In this latter case we will typical indicate the function from which it is derived using the subscript in Σ_f. In summary, the sigma matrix of a given function f is simply the column accumulator of the image matrix of f, i.e., $\Sigma_f = \text{ColSum}(F^+)$. This allowing a computation of Σ_f without computing the matrix of elements F^- from which its definition is derived.

Making this recipe algorithmic, we see that to construct the image matrix F^+ a list of lists. Here the sub-lists $F^+[\,*\,, j]$ are columns of a matrix with elements corresponding to $f^j(x_i)$ for each $1 \leq i \leq n$. I.e., $F^+[i, j]$, corresponds to the result of applying f to x_i j times. We call this procedure P_+ .

In pseudocode the procedure P_+ becomes

```
PPlus = [ ]
j=1
for j in range 1 to n
    append column [ ]
    for i in range 1 to n
        append to column $f^j(x_i)$
```

Pseudocode 2.1

Creation of the Image Matrix.

As we did with the image matrix, to construct the ColSum of an $n \times n$ matrix M, we start with an empty list. To this list we append a list corresponding to the rows $\text{ColSum}(M)[i, *]$ each having a j^{th} elements corresponding to a count of the number of occurrences of x_i in the j^{th} column of M. We call this procedure P_{cs} which is the procedure for generating the column sum of a given $n \times n$ matrix.

In pseudocode the procedure P_{cs} becomes

```
ColSum(M) = [ ]
i=1
for each row i in M
        append an empty row i to ColSum(M)
        for j in range 1 to n
                append count($x_i$) to row of ColSum(M)
```

Pseudocode 2.2

Creation of the ColSum Matrix

We need a procedure P_Σ for computing the sigma matrix. This procedure will be composed of two sub-procedures. The first sub-procedure P_+ takes a function f from the set X^X and outputs its image matrix, an $n \times n$ matrix of singletons; i.e., $F^+ \in \{ * \}^{n\times n}$. The second sub-procedure P_{cs} takes as input the image matrix and outputs the sigma matrix, a matrix of non-negative integers $\Sigma_f \in \mathbb{Z}^{+(n\times n)}$. Letting $F^+{}_{X^X}$ be the set of all image matrices, and $\Sigma_{X^X}$the set of all sigma matrices.

$$P_+ : X^X \to F^+{}_{X^X} \;\; s.t. \;\; P_+(f) = F^+ \tag{8}$$

$$P_{\text{cs}} : F^+{}_{X^X} \to \Sigma_{X^X} \;\; s.t. \;\; P_{\text{cs}}(F^+) = \Sigma_f \tag{9}$$

Thus the composite is defined

$$P_\Sigma := (P_{\text{cs}} \circ P_+) : X^X \to F^+{}_{X^X} \to \Sigma_{X^X} \tag{10}$$

$$P_\Sigma(f) = P_{\text{cs}} \circ P_+(f) = \Sigma_f \tag{11}$$

3. Results on Sigma Matrices

3.1 Properties of the sigma matrix. In Section 2 the primary data structures of this paper were defined: the image, preimage and sigma matrices. The image matrix F^+ with entries $f^j(x_i)$ corresponds to the images of domain elements x_i under j application of f. The preimage matrix F^- with entries of the form $f^{-j}(x_i)$, corresponds to the preimage of domain elements x_i, under j applications of f^{-1}. Lastly, the sigma matrix $\Sigma \in (\mathbb{Z}^+)^{n\times n}$ was defined element-wise as having entries corresponding to the size of the preimage matrix entries at the same position. I.e., $\Sigma[i,\ j] := |f^{-j}(x_i)| = |F^{-}[i,\ j]|$.

Having now defined the sigma matrix and shown its construction from a column accumulation of the image matrix, we move on to describing properties of Σ. We will refer to a property of Σ which holds for all sigma matrices, independent of the particular function used to build it, as a *universal property* of sigma matrices Σ. Likewise a property which depends on properties of a particular function f as a *local property* of the sigma matrix Σ_f. For example, we will prove that the sum of all elements in any column of a sigma matrix is invariant under f. This invariance is thus a universal property of these matrices. Contrastingly, we will prove that any constant function's sigma matrix will have a row of all n' *s*. Since this holds for all constants, but not all functions in X^X, this is a local property of such matrices.

3.2 Universal Properties of Sigma Matrices. We start with some universal properties of sigma matrices that will be useful going forward. These are properties of all sigma matrices and thus tell us something about the preimage set sizes of arbitrary endofunctions under iterated inverse. These properties will be useful in specifying the structure of sigma matrices and in confirming a correct computation of Σ for a given f.

Proposition 3.1: *All entries of sigma matrices are non-negative integers.*

Proof: Since by the definition of a sigma matrix $\Sigma[i,\ j] := |F^{-}[i,\ j]| := |f^{-j}(x_i)|$, the entries $\sigma_{(i,j)}$ correspond to set sizes of the preimage matrix. Sets are either empty or non-empty. If a set is empty its size is zero. If a set is non-empty its size is non-zero and positive. Thus $\sigma_{(i,j)} \geq 0$ for all $1 \leq i, j \leq n$ since $|f^{-j}(x_i)| \geq 0$ for both empty and non-empty sets alike. This holds for any sigma matrix and thus is a universal property of these matrices.

■

Proposition 3.2 : *If some row element of a sigma matrix is zero , then all remaining elements of that row are also zero.*

Proof. Suppose row i has entry $\sigma_{(i,k)} = 0$ in its k^{th} position. By definition $\sigma_{(i,k)} = \Sigma[i,\ k] = |\,f^{-k}(x_i)|$ and so the associated preimage set $f^{-k}(x_i) = \{x : f^k(x) = x_i\}$ must be empty as $\sigma_{(i,k)} = 0$ tells us that it has no elements in it. If $f^{-k}(x_i)$ is empty so is its preimage set $f^{-(k+1)}(x_i) = \{x : f(x) \in f^{-k}(x_i)\}$. This set must be empty as $\{x : f(x) \in f^{-k}(x_i)\}$ becomes $\{x : f(x) \in \{\}\}$ when $f^{-k}(x_i)$ is empty. Recognizing that $|f^{-(k+1)}(x_i)| = \sigma_{(i,k+1)} = 0$, we see that any matrix element following a zero in the i^{th} row must also be zero.

■

Theorem 3.3. *Each column of a sigma matrix is a partition of n*

Proof: The element $\sigma_{(i,j)}$ of the sigma matrix is constructed using column accumulation of the image matrix. That is, the sigma matrix is defined by $\Sigma_f = \text{Colsum}\big(F^+ : X\big) := \text{Colsum}\big(f^{[1,n]}(X)\big)$ where the elements of F^+ are of the form $f^j(x_i)$. The image of any $x \in X$ under j applications of f is always a singleton since f is a function. Since the forward image matrix F^+ is of type $\{\,*\,\}^{n\times n}$ it has n rows of singletons. Thus, we count a total of n elements populating any particular column $F^+[\,*\,, j]$ and hence any column of the sigma matrix reflects this sum.

Corollary 3.4. *Every sigma matrix Σ_f has n as an eigenvalue.*

Proof: By Theorem 3.3 above, the sum of each row of a sigma matrix is n. A square matrix with common row-sum has eigenvalue equal to the common row sum. Thus an eigenvalue of any sigma matrix is $\big|\text{dom}(f)\big| = n$

■

Corollary 3.5. *The sum of all elements of a sigma matrix is n^2.*

Proof: Theorem 3.5 gives each column sum of n. Since Σ has n columns, it follows that the matrix has a total sum of n^2.

■

Proposition 3.6. *Let $Y = \{y_1, y_2, \ldots, y_j, \ldots y_n\}$ be a system of n equations formed from the columns of a sigma matrix Σ. For example, $y_4 := a_1 + \sum_{i=1}^{n} a_i\, x^i$ where $a_i = \sigma_{(i,4)}$. All polynmials in Y pass through (1, n).*

Proof: Examine the j^{th} equation corresponding to the j^{th} column of the sigma matrix: $y_j = a_1 + a_2 * x + a_3 * x^2 + \ldots + a_n * x^n$. The coefficients are taken from the entries of column j. Letting

$x = 1$ gives $y_j(1) = a_1 + a_2 + ... + a_n$. By Theorem 3.5, the sum of any column is n and so it must be that the sum of all coefficients $\sum_{i=1}^{n} a_i = n$. So the curve y_j passes through $(1, n)$. This argument holds for each column. Thus all the polynomials formed from the columns of a sigma matrix pass through the point $(1, n) \in \mathbb{R}^2$.

■

3.3 Some Local Properties of Sigma Matrices. We now examine some of the properties of sigma matrices for certain types of functions. In particular we are interested in the properties of sigma matrices of the constant function and the bijection. Recalling that by (B5) we require that any complexity measure developed take into account the contrast between these functions.

Proposition 3.7. If f is the constant function on n elements, then Σ_f has a row of all *n's.*

Proof. If f is constant, then there is some c such that for all x, $f(x) = c$. Thus in the graph of f, vertex c has in-degree n and all others have in-degree 0. Recall that j corresponds to the column number and i the row associated with the i^{th} domain element. Without loss of generality assume the first row of Σ_f corresponds to the fixed point of f, i.e., $x_1 = c$. To verify that indeed the first element of such a row is n, we only need to show that the first inverse of the fixed point contains all points of X. Because f is the constant function, the first row element $\sigma_{(1,1)} = |f^{-1}(x_1 = c)| = |X| = n$. Now suppose the j^{th} entry of the row is also n, then the $(j+1)^{th}$ element in row is given by $\sigma_{(1,j+1)} = f^{-(j+1)}(c) = f^{-1}\left(f^{-j}(c)\right)$. Since f is constant and c the fixed point, $f^{-j}(c) = \{x : f^{j}(x) = c\} = X$ thus $f^{-1}\left(f^{-j}(c)\right) = f^{-1}(X)$. Since $c \in X$ and $|f^{-1}(c)| = |X| = n$ we have that $\sigma_{(1,j+1)} = n$.

■

Proposition 3.8. *If f is a bijection then all elements of Σ_f are 1's.*

Proof. For f to be a bijection is for its elements to have uniquely defined first inverses. Thus $f^{-1}(x_i)$ is a singleton for any x_i. I.e, $\sigma_{(i,1)} = 1$. Suppose now that $\sigma_{(i,j)} = |f^{-j}(x_i)| = 1$ then since this is a bijection, there is a unique $f^{-(j+1)}(x)$ so that $f\left(f^{-(j+1)}(x)\right) = f^{-j}(x)$. The set $f^{-(j+1)}(x_i)$ is also a singleton and thus $\sigma_{(i,j+1)} = |f^{-(j+1)}(x)| = 1$ and thus we have shown inductively that for any x_i and any j, $|f^{-j}(x_i)|$ so $\sigma_{(i,j)} = 1$ for any $i, j \leq n$.

■

Example 3.9. Consider $f = \{(a, b), (b, a), (c, c)\}$ and $g = \{(a, a), (b, b), (c, c)\}$

The preimage, forward image, and sigma matrices of f and g are given by

$$F^- = \begin{pmatrix} b & a & b \\ a & b & a \\ c & c & c \end{pmatrix} \quad F^+ = \begin{pmatrix} b & a & b \\ a & b & a \\ c & c & c \end{pmatrix} \quad \Sigma_f = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$G^- = \begin{pmatrix} a & a & a \\ b & b & b \\ c & c & c \end{pmatrix} \quad G^+ = \begin{pmatrix} a & a & a \\ b & b & b \\ c & c & c \end{pmatrix} \quad \Sigma_g = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

These two functions have identical inverse set structures: they are both bijections and thus each element has a unique inverse element. While f has a loop from b to a to b and a self-loop at c, there is no uncertainty about a predecessor. This also holds for g: for this function each element is a self-loop and thus also uniquely determined. Despite the fact that these functions have different inverse structures, they are the same with respect to preimage set *sizes.*

Example 3.10. Sigma Matrices of Constants and Bijections.

If we compare the bijection $f = \{(a, b), (b, a), (c, c)\}$ from above to the constant function $c = \{(a, a), (b, a), (c, a)\}$, we see that we have quite different preimage structures. We see a total consolidation of elements to the first row in Σ_c and the matrix-wide distribution of elements in a bijection as in Σ_g.

$$\Sigma_c = \begin{pmatrix} 3 & 3 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \Sigma_g = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Corollary 3.11. *The constant function has $n-1$ rows of all 0's.*

Proof: By Proposition 3.6 the constant has a row of all n's, and by Corollary 3.4, the sum of all elements of a sigma matrix is n^2. However a row of length n, of all n's, has a sum of n^2. Since no values of a sigma matrix are negative the remaining entries must all be zeros.

■

4. Graphs and Sigma Matrices.

4.1 A Brief Review of Graphs. In the previous section we proved key facts about sigma matrices. For example, we showed that the sum of the entries of any column of a sigma matrix is n. Since every sigma matrix is $n \times n$, we also therefore showed that the sum of all its entries is n^2. Certain functions result in wildly different sigma matrices. For example, the bijections and the constant function have radically different matrices. The constant function has its non-zero entries condensed in a single row of all n's whereas the bijections evenly distribute such entries into all possible coordinates thus having 1's for all entries $\sigma_{(i,j)}$. Next, we turn to an exploration of the relationship between the sigma matrix of f and the graph $G(f)$. The world of finite graphs provides a rich and visual model of finite functions which are easily represented by edges corresponding to the ordered pairs making up function elements. While sigma matrices tell us only about the sizes of preimage sets and not the elements themselves, they nonetheless tell us something about the structure of the associated graph.

We now present a basic vocabulary for talking about graphs associated with functions. Since functions are made up of ordered pairs we will find that a key structure is know as the digraph.

Definition 4.1 : A *directed graph* G as consisting of a finite, nonempty set V of *vertices* and a set E of ordered pairs of vertices from V. The members of E are called (directed) *edges* where V and E are called the *vertex set* and *edge set* respectively. That is, a digraph is an ordered pair of the vertex and edge sets, i.e., $G := (V, E)$.

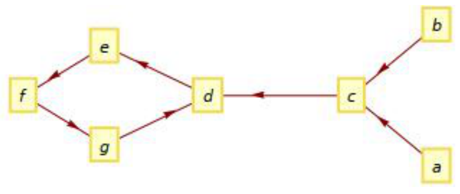
Visually we represent any function $f : X \to X$ as a digraph $G(f)$ by a set of points and arrows. The points correspond to vertices and the arrows to the directed edges. For each given edge $(x, y) \in E$ we associate an arrow given by the projection of the first and second coordinates of the edge pair (x, y).

In a directed graph we often speak about the *in degree* and *out degree* of a vertex: the number of edges directed towards v, notated $\deg_{\text{in}}(v)$, or those directed away from v, $\deg_{\text{out}}(v)$. Since f will be a function, the number of arrows leaving any vertex v is one and thus so is its out-degree. We define the *neighborhood* of a vertex v as all vertices adjacent to v, written as $\text{Nbd}(v)$. Notice that $|\text{Nbd}(v)| = \deg_{\text{in}}(v) + \deg_{\text{out}}(v)$. Finally, any vertex with $|\text{Nbd}(v)| = 1$ is called a *leaf.*

Example 4.2 Let h be represented by its ordered pairs as follows

$$h = \{(a, c), (b, c), (c, d), (d, e), (e, f), (f, g), (g, d)\}.$$

We find the associated graph $G(h) = (V, E)$ and exhibit its vertex set, edge set, and graphical representation. Here the set of all possible vertices are all first and second elements of the elements of h. The *vertex* and *edge sets* are given by $V = \{a, b, c, d, e, f, g\}$ and $E = \{(a, c), (b, c), (c, d), (d, e), (e, f), (f, g), (g, d)\}$. Thus $G(h) = (V, E)$ with the aforementioned V and E. In this example $\deg_{\text{in}}(d) = 2$ since $f(c) = d$ and $f(g) = d$ and $\deg_{\text{out}}(d) = 1$ as f is functional. Here, $\text{Nbd}(d) = \{g, e, c\}$ where $|\text{Nbd}(d)| = |\{g, c\}| + |\{e\}| = 3$. The leaves of G below are vertices b, and a.



$G(h)$

Figure 4.1 A Cyclic Graph of Order 7

The graph above corresponds to the function in Example 4.2. There is a one-to-one correspondence between the domain of h and the vertex set V —— as well as between the set of ordered pairs making up the function, $X \times X$ and the edge set E. Sometimes we will want to verify that a given graph G is indeed *functional, i.e,* that there exists a function f whose associated digraph is G. We do not need to explicitly construct the function associated with G, rather we will simply check the graph for the properties of a functional relation. That is, we check that for an any element $v \in V$ if (v, v') and (v, v'') are in $E(G)$ then $v' = v''$. Since we are developing the theory of functional preimage complexity, all graphs discussed correspond to functions $f : X \to X$ and are thus functional.

Recall that in Section 2 we showed that to build the sigma matrix of a function does not require a calculation of the preimage matrix F^-; rather, we only need calculate the forward image matrix F^+ and take a column accumulation of this matrix. For example, we can calculate the sigma matrix of a function f given its (functional) graph G. We do this for the graph in Figure 4.1.

The first row of F^+ below, corresponding to $x_1 = a$, is found by calculating $[f(a),\ f^2(a),\ \ldots,\ f^{n=7}(a)]$, likewise the last row of Σ_f which here corresponds to $x_7 = g$ is found by calculating the number of $x_7 = g$ in each column of F^+ : there is one g in columns 1, 2 and 3; two in column 4, and so on. Thus from the functional graph G in figure 4.1 we produce the following:

$$F^+ = \begin{pmatrix} c & d & e & f & g & d & e \\ c & d & e & f & g & d & e \\ d & e & f & g & d & e & f \\ e & f & g & d & e & f & g \\ f & g & d & e & f & g & d \\ g & d & e & f & g & d & e \\ d & e & f & g & d & e & f \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 1 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 & 3 & 1 & 1 \end{pmatrix}$$

A single component graph which is acyclic (has no loops) is called a *tree.* We make the following definition as we will use these conditions often:

Definition 4.3 If $f : X \to X$ and then its graph G is a *(*functional*) tree with root* v_0 when

1. The cycle of G is a self loop $(v_0, v_0) \in E$ corresponding to the fixed point x_0 of f so that $f(x_0) = x_0$.
2. G is connected.

Example 4.4 Let $h = \{(8, 6), (7, 4), (6, 4), (5, 4), (4, 2), (3, 2), (2, 1), (1, 1)\}$.

We have a function h whose graph has no loops other than the self-loop on vertex 1. Thus we have a functional tree of order 8 with root $1 \in V(G(h))$ due to fixed point $(1, 1) \in E(G(h))$. Since h is a function, $\deg_{\text{out}}(v) = 1$ for all $v \in V$. The vertex with the largest in-degree is "4" with $\deg_{\text{in}}(4) = 3$, where

$\left|\text{Nbd}(\text{"4"})\right| = 4$ noting that indeed $|\text{Nbd}(4)| = \deg_{\text{in}}(4) + \deg_{\text{out}}(4)$. The set of leaves $L_G \subset V$ is given by $L_G = \{7, 8, 5, 3\}$ as $|\text{Nbd}(v)| = 1$ for all $v \in L_G$. Since h is functional, if $|\text{Nbd}(v)| = 1$ then $\deg_{\text{in}}(v) = 0$. Calculation of the H^+ and Σ_h would be done as in Example 5.1.

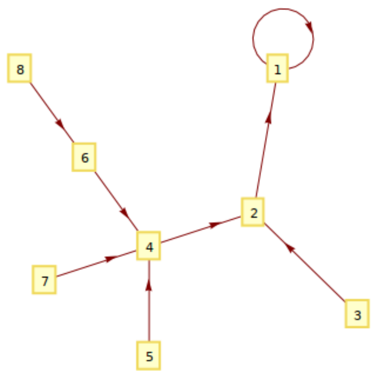


Figure 4.2 A Functional Tree of Order 8

Definition 4.5 A non-empty directed trail in which the first and last vertices are repeated, for example $(v_1, v_2, \ldots, v_n, v_1)$, where there is only one repeated vertex, v_1, is a *directed cycle.*

Example 4.6 The following graph, considering only the black edges $\{(0, 1), (0, 2), (1, 3), (3, 4)\}$ is a directed acyclic graph. However when adding the red edges $\{(2, 0), (3, 0), (4, 4)\}$ we have three cycles $c_1 = (0, 2, 0)$, $c_2 = (0, 1, 3, 0)$, $c_3 = (4, 4)$.

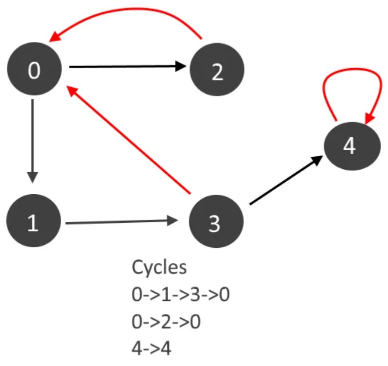


Figure 4.3. A Directed Acyclic (all black edges) and Cyclic (with red) Graph.

4.2 Results on Graphs and Sigma Matrices. The values of the sigma matrix encode information about the graph the function or functions corresponding to that matrix. One important property of graphs is cycle detection and determination of cycle membership. These properties are particularly useful for subsequent proofs which give us further insight into the associated graph G for functions with a particular sigma matrix.

Proposition 4.7 If x_i is a cycle element of $G(f)$ then all elements of the i^{th} row of Σ_f are positive and non-zero.

Proof: If some vertex x_i of $G(f)$ is in a cycle $C \subseteq G(f)$, x_i is either the intersection of some tree $T \subseteq G(f)$ entering the cycle, or is purely a cycle element. Suppose for the first case that x_i is not an element of some tree. Then for any given tree T, $x_i \notin T \cap C$, and thus x_i has only one predecessor: $|f^{-1}(x_i)| = 1$. In the second case, where $x \in T \cap C$ for some tree $T \subseteq G$, then there exists at least two distinct vertices y_1, y_2 of $G(f)$ where $\{y_1, y_2\} \subseteq f^{-1}(x)$. One of these vertices is the predecessor of x_i in the cycle C and the other is the predecessor in the tree T. Thus, in this case we have $|f^{-1}(x_i)| > 1$. In short, if x_i is a cycle element, there is a non-empty preimage $f^{-j}(x_i)$ for any natural number j and thus $\Sigma[i, j] = |f^{-j}(x_i)| \geq 1$ for any j, all elements of the i^{th}row are non-zero.

■

Proposition 4.8 If x_i is not a cycle element of $G(f)$ then some element of the i^{th} row will be zero.

Proof: If x_i is not part of a cycle, it is part of a tree. At minimum, every functional digraph on n vertices contains a cycle of length 1. Thus, at maximum, the number of vertices not part of a cycle is $n - 1$. Therefore, the maximum distance from some x_i to a leaf is $n - 1$. Lastly, for any leaf x_l, $f^{-1}(x_l) = \emptyset$, thus since each row has has n elements, there is some $j \leq n - 1$ so that $\sigma_{(i,j)} = 0$.

■

Proposition 4.9 Let $f : X \rightarrow X$. Denote by L_G the set of vertices of G which are leaves. $|L_G| = q$ if and only if there are q rows of all zeros in Σ_f.

Proof: Thus the first column of Σ_f has entries $\sigma_{(i,1)} = \{x : f^{-1}(x) = x_i\}$ for $1 \leq i \leq n$. If x_i is a leaf in $G(f)$ then $f^{-1}(x_i) = \emptyset$ and thus $|f^{-1}(x_i)| = 0$. If x_i is not a leaf of $G(f)$ then there is some $x : f(x) = x_i$ and thus $f^{-1}(x_i) \neq \emptyset$ and so $|f^{-1}(x_i)| = \sigma_{(i,1)} \neq 0$.

By proposition 3.2, if $\sigma_{(i,j)} = 0$ then $\sigma_{(i,k)} = 0$ for all $k > j$. So for each $x_i \in L_G$, the i^{th} row of Σ_f is all zeros and for each $x_i \notin L_G$ we do not have an i^{th} row of all zeros. Therefore if $|L_G| = q$ then we have q rows of all zeros.

Conversely if we have q rows of all zeros then we have q rows where $\sigma_{(i,1)} = 0$. By definition $\sigma_{(i,1)} = 0$ gives us that $|f^{-1}(x_i)| = 0$ and thus $f^{-1}(x_i) = \emptyset$. However $f^{-1}(x_i) = \emptyset$ only when $x_i \in L_G$ and therefore $|L_G| = q$.

■

Proposition 4.10 If the columns of Σ_f converge to n then $G(f)$ is a functional tree.

Proof: By Proposition 4.2, when x_i is not a cycle element then for some $k < n$, $\sigma_{(i,k)} = 0$. So if no element of the i^{th} row is zero, x_i is a cycle element. Now since each column of Σ_f partitions n, if $\sigma_{(i,n)} = n$ then all remaining entries of the n^{th} column must be zero. I.e., we have $\sigma_{(k,n)} = 0$ for $k \neq i$, thus the only cycle element is x_i. The only functions corresponding to (x_i, x_i) are functional trees.

■

5. Preimage Complexity Functions

5.1 In the last section, we showed how structural information about a function, and its associated directed graph, could be extracted from the sigma matrices. Such structural facts included cycle detection, information about level sets and vertex height. At the core of the chapter we define a preimage complexity function, a generalization of what we discovered to be relevant with respect preimage complexity specification. We exhibit the flexibility of the preimage complexity function by giving three complete examples of such preimage complexity functions and examine how each views complexity differently. In essence we prove that our definition of preimage complexity fulfills our goals and benchmarks outlined in section 1. In the final section we will discuss the set of all sigma matrices which may be generated on the set of X^X and provide a complete analysis of the associated sigma matrices, complexities under a chosen preimage complexity function.

5.2 Desired Characteristics of Preimage Complexity Function. The first of three requirements we placed on any definition of a preimage complexity function corresponds to benchmark B3 in section 1.2. That is, we require that any measure of the preimage complexity of a function f must, as the terminology suggests, be a function of the preimage data structures. The definition of a preimage complexity function $H : X^X \to S$ will obey this requirement as there will be defined an auxiliary function $K : \Sigma_X \to S$ taking sigma matrices to values of a finite set in such a way that if two sigma matrices have equal images under K then their generating functions will likewise have equal values under H. Thus, H will partition all functions $f : X \to X$ by way of their value under K which is a function of the preimage data structure - the sigma matrix.

Our second requirement, (B4) of the preimage complexity function is that it be flexible and informative. While equality detection among sigma matrices is important, the complexity function should distinguish properties that two such matrices have in common despite being entry-wise different. For example, we may be interested in the number of 0's in a sigma matrix. There are multiple matrices with an equal number of zeros, there is at least one pair f, g where $\Sigma_f \neq \Sigma_g$ while $K(\Sigma_f) = K(\Sigma_g)$. We are extracting useful information from the recognition of commonalities between f and g despite their distinct sigma matrices. Finally, in addition to extending our view of functions beyond mere comparison of their sigma matrices, we occasionally wish to speak about the relative complexity of two functions. To this point, inequality in S of matrices under K simply meant their respective function complexities were not the same - as they would each inhabit a different block of the partition of X^X by $\sim_H$. Since $H(f) \in S$, and S may or may not be ordered we can only conclude that the above statements are meaningless except to communicate the fact that $H(f) = 0 \neq 6 = H(g)$. We will address this shortly after the definition.

To satisfy B5 any complexity measure based on the preimage structures thus far developed should take into account the natural dichotomy between the functions which are bijections and the constants. This contrast is evident on both a structural and informational level. Structurally, the difference between the sigma matrices of bijections and of the constant functions is stark. In particular, the distribution of the n^2 values of the sigma matrix is illustrative. In the constant function we have a row of all n' s like any sigma matrix each column is a partition of $|X| = n$ however the row distribution of integers summing to n^2 is an extreme case for both. In the bijection we have a matrix of values evenly distributed: each row $\Sigma[i, *]$ and each column $\Sigma[*, j]$ of its sigma matrix is a partition of n. That is

$$\sum_i \sigma_{\mathrm{ij}} = \sum_j \sigma_{\mathrm{ij}} = n \text{ since } \sigma_{(i,j)} = 1 \text{ for all } 1 \leq i, j \leq n.$$

In addition to the distribution of values in the sigma matrices there are information-theoretic differences between the constants $c: X \longrightarrow X$ and the bijections $\alpha: X \longrightarrow X$. The sigma matrix is a representation of the size of predecessor sets at varying inverse depths. One natural question is to determine the set of possible outcomes when computing applying f then f^{-1} for any $x \in X$. Note that while f^{-1} may be empty, the set $f^{-1}(f(x))$ is always inhabited and thus is a way to measure the number of siblings of x or measure the number of alternative paths from x to $f(x)$. In essence, this question is tied to a quantification of determinism. If, for example, the set of possible elements in the set $f^{-1}(f(x))$ is large then the relative bijectivity of f at $y = f(x)$ is low: many elements x have image $y = f(x)$, whereas if $f^{-1}(f(x))$ is a small set then only a small number of elements could be ancestors to y. Given a bijection α and an arbitrary element $x_i \in X$, $\alpha^{-1}(\alpha(x_i)) = \{x_i\}$, a singleton, and thus α^{-1} is uniquely determined. Information is preserved. In contrast, letting $c: X \longrightarrow X$ be a constant, given any $x_i \in X$, $c^{-1}(c(x_i)) = \{x_1, x_2, \ldots, x_i, \ldots x_n\} = X$ and thus all information is destroyed.

We formalize this by providing a natural ordering on the set of sigma matrices Σ_{X^X} which matches our intuition.

5.3 The Composition Poset of Sigma Matrices. Recall that we desire any measure of preimage complexity to respect the natural dichotomy between the preimage structure of the constant and the bijections as reflected by the sigma matrices. To highlight the visual form any non-zero entry of a sigma matrix are indicated by black squares and zeros by white squares. In the diagram below we show the visual form for the sigma matrix of three functions: a bijection, a tree, and a constant.

$$\begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{pmatrix} \begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \square & \square \\ \blacksquare & \square & \square & \square \\ \square & \square & \square & \square \end{pmatrix} \begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{pmatrix}$$

Figure 5.1

Non-Zero Entries of Σ_f for f a bijection, tree and constant respectively.

Following our demands that for a bijection α and constant c, $H(\alpha) \neq H(f)$ and $H(f) \neq H(c)$ for any f, we wish to find a poset structure on the sigma matrices so that Σ_α and Σ_c are at opposing extremes of such an ordering. The set Σ_X of all sigma matrices for the endofunctions on a set X may be endowed with a poset structure in a natural way which respects the above dichotomy.

We aim to traverse the set of these sigma matrices via composition of the underlying functions noting that given $f \circ g = h$ the size of the image set of h must be smaller or equal to the size of the image set of g for any f. At the extreme, a maximal restriction of an endofunction's image set to a singleton corresponds to a maximal lack of injectivity, and thus to the constant function. This is reflected by the fact that since composition yields a consolidation of the image set, for any function f and for a constant c, $f \circ c$ must remain a constant. At the other end of our spectrum we find a similar situation. The function with a maximal image set is a bijection as for any $\alpha : X \to X$, $\mathrm{Im}(\alpha) = X$. Given $f \circ \alpha = h$ where g is a bijection, the application of $f$can only serve to restrict (or maintain) the image of f relative to α.

We summarize this as follows.

1. If $f \circ g = h$ then $\left|\mathrm{Im}(h)\right| \leq |\mathrm{Im}(g)|$ for any g.
2. If α is a bijection then for any h there is always some f so that $f \circ \alpha = h$
3. If c is a constant and $g \neq c$ then there does *not exist* an f so that $f \circ c = g$.

Using the above simple properties we define the poset of sigma matrices as follows.

Definition 5.1: Given two functions $f_1, f_2 : X \to X$ we will define the *poset of sigma matrices* $(\Sigma_X, \leq)$ in the following way: we say that $\Sigma_{f_1} \leq \Sigma_{f_2}$ *iff* there is some $g \in X^X$so that $f_2 = g \circ f_1$. Similarly we write $\Sigma_{f_1} \sim \Sigma_{f_2}$ *iff* there is some g_1 so that $f_2 = g_1 \circ f_1$ and some g_2 so that $f_1 = g_2 \circ f_2$

We can easily see the following which nicely follows our notions developed above.

Property 5.2: $\Sigma_\alpha \leq \Sigma_f$ for all f and α a bijection. Also, $\Sigma_f \leq \Sigma_c$ for all f and c a constant.

Given any bijection α and any function f there is some g so that $g \circ \alpha = f$, thus $\Sigma_\alpha \leq \Sigma_f$. Likewise, since for any f, $c \circ f = c$ we have that $\Sigma_f \leq \Sigma_c$.

Before defining our preimage complexity function, recall that procedure P_Σ calculates the sigma matrix of f by way of two sub-procedures: P_+ first calculates the $n \times n$ image matrix F^+ of a function f, then P_{cs} calculates the sigma matrix Σ_f from F^+, the result of P_+. In short, $P_\Sigma : X^X \to \Sigma_X$ where $P_\Sigma = P_{\text{cs}} \circ P_+$. We now present the definition of a preimage complexity function in satisfaction of our benchmarks B3-5.

Definition 5.3. Let $H : X^X \to S$ be the composite $H := K \circ P_\Sigma$ where $P_\Sigma : X^X \to \Sigma_{X^X}$ builds a function's sigma matrix and $K : \Sigma_{X^X} \to S$ assigns the matrix a value in a finite set S. If the following holds then H is a *preimage complexity function.*

1. Equality of sigma matrices implies equality under H
$$\text{If } \Sigma_f = \Sigma_g \text{ then } K(f) = K(g)$$
2. Equality of sigma matrices under K implies the associated functions are equal under H.
$$\text{For all } f,\ g \in P_\Sigma^{-1} \circ K^{-1}(s),\quad H(f) = H(g)$$
3. Any bijection α and any constant c have different values under H.
$$H(\alpha) \neq H(c)$$

In addition to the properties of a preimage complexity function from Definition 5.1 above, we may wish to give meaning to statements like "the preimage complexity of f is less than that of g." In such a case, we define a preimage complexity function with order.

Definition 5.4 A *preimage complexity function with order*, H, satisfies all the properties of Definition 5.1 and additionally requires that the image of matrices under K are real numbers where their relative magnitude is respected by H. That is, for any f_1 with sigma matrix Σ_1, and f_2 with sigma matrix Σ_2, $K : \Sigma_X \longrightarrow S \subset \mathbb{R}$ and if $K(\Sigma_1) = s_1 \leq s_2 = K(\Sigma_2)$ then $H(f) \leq H(g)$.

5.3 Example Preimage Complexity Functions. Per our discussion in section 5.2, preimage complexity functions are to be flexible and useful: we will want PCFs which can finely and coarsely partition X^X according to our needs. The two examples below illustrate the opposing ends of this spectrum: the discrete (preimage) complexity function H_d and the indiscrete complexity function H_X. The discrete preimage complexity function, the finest partitioning possible $P_{\sim_H}$ of X^X by H, maximizes the number of blocks of X^X partitioned by a preimage complexity function by demanding that no commonality other than equality may be considered between two sigma matrices. In other words, if $\Sigma_f \neq \Sigma_g$ then there is no other property of the matrices that could make the image of their parent functions f, g equal under H. Contrastingly, the indiscrete complexity function H_X results in the partition of X^X with the fewest blocks. In this complexity function all sigma matrices are considered equal under K unless they are the sigma matrices of bijections or constants -- in which case they are given their own value under H. This is the coarsest possible partitioning of X^X by a preimage complexity function.

Example 5.5 *The discrete preimage complexity function, H_d*

The PCF which most finely partitions X^X is the function that requires equality of sigma matrices to partition their parent functions in the same block. This is an extension of Definition 5.1 to bidirectionality

$$\Sigma_f = \Sigma_g \ \textit{iff} \ H(f) = H(g).$$

Let S be the set of all sigma matrices of the functions in X^X, i.e., $S = \Sigma_X$ giving $K : \Sigma_X \longrightarrow \Sigma_X \ s.t.$

$$H(f) := K(P_\Sigma(f)) \ s.t \, K(\Sigma) = \Sigma$$

Proposition 5.6. *H_d is a preimage complexity function.*

Proof: Firstly, if $\Sigma_f = \Sigma_g$ then K as defined above gives us that $K(\Sigma_f) = K(\Sigma_g)$ satisfying part one of the definition. Secondly since $K(\Sigma_f) = s = K(\Sigma_g)$ only when $\Sigma_f = \Sigma_g$ then $K(\Sigma_f) = s = K(\Sigma_g) \Longrightarrow \Sigma_f = \Sigma_g$ and thus $H(f) = H(g)$ by (1). Lastly if α is a bijection and c the constant then $\Sigma_\alpha \neq \Sigma_c$ by section 3 and thus by $K(\Sigma) = \Sigma$ above, $K(\Sigma_\alpha) = \Sigma_\alpha \neq \Sigma_c = K(\Sigma_c)$. Thus by definition of $H_d(f) := K(P_\Sigma(f))$ we have $H(\alpha) \neq H(c)$ satisfying (3). Hence H as defined is indeed a PCF which we call the discrete PCF and denote it H_d.

■

Now, on the opposite side of the spectrum is the preimage complexity measure with resulting in the coarsest partitioning of X^X, the indiscrete PCF.

Example 5.7. *The indiscrete PCF.*

The indiscrete PCF on X, denoted H_X, is the function which partitions together all functions allowable by the definition. Thus in the indiscrete PCF we want $H(\alpha) \neq H(f) = H(g) \neq H(c)$ for any bijection α, a constant function c, and all other functions neither constant nor bijective. Let $S = \{s_\alpha, s_c, s\}$ and define $K : \Sigma_X \to S$ and H by

$$K_X(\Sigma) = \begin{cases} s_\alpha & \text{if } \Sigma = \Sigma_\alpha \\ s_c & \text{if } \Sigma = \Sigma_c \\ s & \text{otherwise} \end{cases},$$

$$\text{where } \Sigma_\alpha[i,j] = 1\,, \quad \Sigma_c[i,j] = \begin{cases} n & i = 1 \\ 0 & i \neq 1 \end{cases}$$

Proposition 5.8. *H_X is a preimage complexity function.*

Proof: We satisfy definition 5.1 part (1) since if $\Sigma_1 = \Sigma_2$ then $K_X(\Sigma_1) = K_X(\Sigma_2)$. We also satisfy 5.1.2: Let α_1 and α_2 be bijections. Their sigma matrices are matrices of all 1's and thus have the property that $\Sigma[i,j] = 1$ for all $1 \leq i, j \leq n$. So,

$$P_\Sigma(\alpha_1) = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} = P_\Sigma(g) \text{ and thus } K(\Sigma_{\alpha_1}) = K\left(\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}\right) = s_\alpha = K(\Sigma_{\alpha_2}).$$

Equality of α_1 and α_2 under H follows directly from the definition $H_X(\alpha_1) := K(P_\Sigma(\alpha_1))$ since

$$H(\alpha_1) = K(P_\Sigma(\alpha_1)) = K\left(\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}\right) = s_\alpha,$$

$$H(\alpha_2) = K(P_\Sigma(\alpha_2)) = K\left(\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}\right) = s_\alpha.$$

so indeed $H(f) = H(g)$. Likewise for any constants c_1, c_2,

$$P_\Sigma(c_1) = \begin{pmatrix} n & \cdots & n \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \end{pmatrix} = \Sigma_c = P_\Sigma(c_2) \text{ and so } K(\Sigma_{c_1}) = K\left(\begin{pmatrix} n & \cdots & n \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \end{pmatrix}\right) = s_c = K(\Sigma_{c_2}).$$

similarly $H_X(f) := K(P_\Sigma(f))$ gives

$$H(c_1) = K(P_\Sigma(c_1)) = K\left(\begin{pmatrix} n & \cdots & n \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \end{pmatrix}\right) = s_c,$$

$$H(c_2) = K(P_\Sigma(c_2)) = K\left(\begin{pmatrix} n & \cdots & n \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \end{pmatrix}\right) = s_c,$$

so $H(c_1) = H(c_2)$ thus satisfying part two of our definition. This property of equivalence under H must hold for all other functions $f : X \to X$ as well. Letting $P_\Sigma(f_1) := \Sigma_1$ and $P_\Sigma(f_2) := \Sigma_2$, if f_1 and f_2 are neither bijections nor constants then Σ_f is neither a matrix of all ones or a matrix of all n's in the first row. I.e., Σ_f is neither Σ_α nor Σ_c and therefore $K(\Sigma_1) = K(\Sigma_2) = s$. So under H

$$H(f_1) = K(P_\Sigma(f_1)) = K(\Sigma_1) = s,$$

and

$$H(f_2) = K(P_\Sigma(f_2)) = K(\Sigma_2) = s,$$

Lastly, in accordance with (3) the bijections and constants must have different values under H. Here,

$$H(\alpha) = s_\alpha \neq s_c = H(c).$$

This proves our claim that indeed H_X is a preimage complexity function.

■

Proposition 5.9. *There is no H which trivially partitions all of X^X into one block.*

Proof: Suppose there were, then for any $f, g \in X^X$, $f \in [g]_{\sim_H}$ and $H(f) = H(g)$. But since the bijections and constants would then be in the same and only block and hence would have the same values under H contradicting part 3 of our definition.

■

In addition to the properties of a preimage complexity function from definition 5.1 above, we may wish to give meaning to statements like "the preimage complexity of f is less than that of g." In such a case, we define a preimage complexity function with order.

Definition 5.10 A *preimage complexity function with order,* $H_\leq$, satisfies all the properties of Definition 5.1 and additionally requires that the image of matrices under K are real numbers where their relative magnitude is respected by H and that the complexity of a bijection and a constant are the minimum and maximum respectively of any functions in X^X. Let $K : \Sigma_X \longrightarrow S \subset \mathbb{R}$ and let c be a constant function and α be a bijection. Then in addition to the properties in definition 5.1, the following must hold

$$K(\Sigma_1) = s_1 \leq s_2 = K(\Sigma_2) \Longrightarrow H(f_1) \leq H(f_2) \text{ for } f_1 \in P_\Sigma^{-1} \circ K^{-1}(s_1) \text{ and } f_2 \in P_\Sigma^{-1} \circ K^{-1}(s_2)$$

$$H(\alpha) \leq H(f) \text{ for all } f \in X^X$$

$$H(f) \leq H(c) \text{ for all } f \in X^X$$

Recall that we can view the constant and the bijection diametrically as destroying and preserving information respectively.

Consider the number of zeros in some sigma matrix Σ_f. As we move from a bijection α to a constant function c, we move from corresponding sigma matrices with n^2 ones, and thus no zeros, to a matrix with $n(n-1)$ zeros. The more zeros trailing a row $\Sigma[i, *]$ the shorter the maximum path from x_i to a leaf and the fewer steps required to step backwards through all predecessors of x_i. One crude measure of this relative consolidation of nodes towards the root, and thus of the function away from bijectivity, is a count of the number of zeros in Σ_f.

Example 5.11. A measure of bijectivity, H_0.

The definition of a PCF is flexible enough to suit many interpretations of relative complexity. If we claim that increased relative bijectivity is decreased complexity then the complexity is proportional to the number of zeros in the sigma matrix: more zeros to corresponds to a higher complexity. To accomplish this, we define a function z and assign a value of $z(\sigma_{(i,j)}) = 1$ if $\sigma_{(i,j)} = 0$ and assign a value $z(\sigma_{(i,j)}) = 1$ otherwise. We then let K sum the number of entries equal to zero.

Let $S = \mathbb{Z}^+$ and and define $K_0 : \Sigma_X \rightarrow \mathbb{Z}^+$ and H by

$$K_0(\Sigma) = \sum_{i=1,j=1}^{n,n} z(\sigma_{(i,j)}) \ : \ z(\sigma_{(i,j)}) = \begin{cases} 1, & \sigma_{(i,j)} = 0 \\ 0, & \sigma_{(i,j)} \neq 0 \end{cases}$$

Proposition 5.12: *H_0 is indeed a PCF with order*

Proof: Clearly if two matrices Σ_1 and Σ_2 are the same, then they have the same number of zeros and thus $K(\Sigma_1) = K(\Sigma_2)$. Next, if two matrices have the same number of zeros, say m, then they are in the set of all sigma matrices with m zeros:

$$K(\Sigma_1) = m = K(\Sigma_2) \text{ then } \Sigma_1, \Sigma_2 \in K^{-1}(m) = \{\Sigma : K(\Sigma) = m\}.$$

If a function f is one such function having a sigma matrix with m zeros then it is in the set $K^{-1}(m)$. We can write this nested set membership as

$$f \in \{f : P_\Sigma(f) \in K^{-1}(m)\} = P_\Sigma^{-1} \circ K^{-1}(m)$$

which is what we wished to show: that for any f_1, f_2 in the above set $H(f_1) = H(f_2)$.
We need to show that when applying H, the bijections and constants have different values. Since we know that any bijection α has no zeros in its sigma matrix and constants have $n(n-1)$ zeros then $H(c) = n(n-1)$. Since $n \geq 2$, $H_0(c) = n(n-1) \geq 2 \neq 0 = H_0(\alpha)$. Thus H_0 is a preimage complexity function.
Lastly, we show that H_0 is a PCF with order. $\mathrm{Cod}(K_0) = \mathbb{Z}^+$ which is a subset of the reals. Suppose $\mathrm{K}(\Sigma_1) = \mathrm{m}_1 \leq \mathrm{m}_2 = \mathrm{K}(\Sigma_2)$ then the count of zeros in Σ_1 is less than the count of zeros in Σ_2. Since $H := K \circ P_\Sigma$ it follows that $H(f_1) \leq H(f_2)$ for all $f_1 \in P_\Sigma^{-1} \circ K^{-1}(m_1)$ and $f_2 \in P_\Sigma^{-1} \circ K^{-1}(m_2)$. Finally the number of zeros in bijections α is zero, and the number of zeros in the constant is $n(n-1)$ thus for all f, $H_0(\alpha) = 0 \leq H_0(f) \leq n(n-1) = H_0(c)$.

■

Suppose for some application we say that a decrease in information preservation is correlated with an increase in complexity and that a decrease in information preservation is correlated with an increase in complexity. In such a case we are going beyond a delineation of equality under H. We are not claiming that $|H(f) - H(g)|$ is meaningful, rather we wish to say that if f is less complex than g then we can write $H(f) \leq H(g)$. Next, if we claim that bijectivity of f is inversely proportional to the number of zeros of Σ_f then if the number of zeros in Σ_f is less than the number of zeros of Σ_g then we need that $H(f) \leq H(g)$. We prove this holds for the extremes of the bijections and the constants.
To prove this holds for the constants and bijections we simply determine the number of zeros in their respective sigma matrices and verify that this is proportional to the value of H. We have a ready-made function to handle the counting of zeros in a sigma matrix, namely K_0. From our work in chapter 3 we know that the constant function has a sigma matrix with $n(n-1)$ zeros and that bijections have a sigma matrix with no zeros. In the example above we found that $K_0(\Sigma_c) = n(n-1) \neq K_0(\Sigma_\alpha) = 0$ implied $H(c) \neq H(\alpha)$, here we can extend this to say

$$K_0(\Sigma_c) = n(n-1) \geq K_0(\Sigma_\alpha) = 0 \Longrightarrow H_0(\alpha) \leq H_0(c).$$

It should be noted that how our auxiliary function K is defined can impact the intuition behind a measurement of complexity. For example, had we defined

$$z(\sigma_{(i,j)}) = \begin{cases} 0, & \sigma_{(i,j)} = 0 \\ 1, & \sigma_{(i,j)} \neq 0 \end{cases},$$

we would be counting the number of non-zero elements and thus a larger value of K_0 would correspond to a lower complexity assignment.

6. The Partition Induced by Preimage Complexity Functions.

In this section we examine the induced partition of endofunctions on X by preimage complexity functions. The importance of the discrete PCF is highlighted and we examine the sequence and size of induced partitions as a function of $\left|\text{dom}(f)\right| = n$. Each preimage complexity function H induces a partition $\mathcal{P}_H$ of X^X. In examples 5.2 and 5.4 we have seen the extremes of coarseness and fineness of such a partitioning under the indiscrete H_X and discrete H_d complexity functions respectively. In the case of the indiscrete function H_X we only have three complexity classes. Preimage complexity functions ultimately rely on the properties of the sigma matrices of all endofunctions on a given set. The more sensitive the function is to changes in the preimage structure, the finer the measure. The PCF which most closely tracks the underlying preimage structure is the discrete measure H_d. Here, if two sigma matrices differ, the values of the associated functions under H differs.

Example 6.1. The partition $\mathcal{P}_{H_0}$ induced by H_0 on $X = \{a, b, c\}$.

Since H_0 partitions X^X by the number of zeros of the function's associated sigma matrix, we count the zeros in each of the above and partition the set of matrices by this count. The range of possible values of f under H_0 is {6, 5, 3, 0} -- an accounting of the number of possible zeros of the sigma matrices in $\Sigma_{\{a,b,c\}}$. If we associate each block of the partition with the value of its members under H_0, i.e., a count of the zeros in the respective sigma matrices, we find that

$$[f]_6 = \left\{f : H(f) = 6\right\} = \left\{f : \Sigma_f = \begin{pmatrix} 3 & 3 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right\}$$

$$[f]_5 = \left\{f : H(f) = 5\right\} = \left\{f : \Sigma_f = \begin{pmatrix} 2 & 3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right\}$$

$$[f]_3 = \left\{f : H(f) = 3\right\} = \left\{f : \Sigma_f \in \left\{\begin{pmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix}\right\}\right\}$$

$$[f]_0 = \{f : H(f) = 0\} = \left\{f : \Sigma_f = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}\right\}$$

So for $n = 3$, $\mathcal{P}_{H_0}(3) = \{[f]_6, [f]_5, [f]_3, [f]_0\}$ and thus $|\mathcal{P}_{H_0}(3)| = 4$.

Example 6.2. The partitions $\mathcal{P}_{H_d}$ induced by H_d on $X = \{a, b\}$ and $\{a, b, c\}$

1. Recall that we defined $H_d(f) = H_d(g)$ if and only if $\Sigma_f = \Sigma_g$. Thus the number of blocks of the partition induced by H_d is the same as the number of unique sigma matrices, $|\mathcal{P}_d| = |\Sigma_{X^X}|$. For $X = \{a, b\}$ we can calculate $|\mathcal{P}_d|$ by looking at all unique sigma matrices on $\{a, b\}^{\{a,b\}}$. This gives us
$$\Sigma_{X^X} = \Sigma_{\{a,b\}^{\{a,b\}}} = \left\{\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 0 & 0 \end{pmatrix}\right\}$$
and so $\left|\mathcal{P}_{H_d}(\{a, b\})\right| = 2$. We write $|\mathcal{P}_{H_d}(2)| = 2$ to mean the number of blocks of the partition $\mathcal{P}_{H_d}$ on two elements.

2. For $X = \{a, b, c\}$ we have $3^3 = 27$ possible functions, the set of which is X^X. We have three constants each with a sigma matrix of all zeros except for a row of all 3's. Next we have six bijections each with a sigma matrix of all ones. Finally, we have eighteen other functions -- six of which correspond to each of three different sigma matrices. Thus the unique set of sigma matrices for $\{a, b, c\}^{\{a,b,c\}}$ are given by the set
$$\Sigma_{X^X} = \Sigma_{\{a,b,c\}} =$$
$$\left\{\begin{pmatrix} 3 & 3 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}\right\},$$
therefore $\left|\mathcal{P}_d(\{a, b, c\})\right| = |\Sigma_X| = 5$. I.e., $|\mathcal{P}_d(3)| = 5$.

We have shown the way in which preimage complexity functions induce partitions of the endofunctions on a given set and given several examples. A natural question to ask is : *given some H for each domain size n, how many equivalence classes are there?*

Definition 6.3. The sequence of partition sizes as a function of $|X| = n$, induced by a preimage complexity function $H_\sim$, will be written $\mathcal{P}_{H_\sim}(n)$.

Example 6.4. We see from the above two examples that we know $\mathcal{P}_d(2) = 2$ and $\mathcal{P}_d(3) = 5$. We have to be careful *not* to assume that $\mathcal{P}_{H_d}(n) = \Sigma(n)$. The only issue here is $\mathcal{P}_d(1)$. While there is a single sigma matrix on X^X where $|X| = 1$, namely a 1×1 matrix with entry 1, there is no induced partition on $X = \{x_1\}$ as the function $f(x_1) = x_1$ is both constant

$$\mathcal{P}_{H_d}(n) = \{\mathcal{P}_d(1), \mathcal{P}_d(2), \mathcal{P}_d(3) \ldots\} = \{0, 2, 5, \ \ldots\}$$

Since the discrete preimage complexity function H_d simply assigns a block to any function with a unique sigma matrix we can use the upcoming question to help us answer produce values in the sequence $\mathcal{P}_d(n)$.

6.2 The Novel Sequence $\Sigma(n)$ The number of sigma matrices for all $f \in X^X$ as a function of domain size n is of great importance to any calculation of a preimage complexity function. Interestingly this sequence appears to be novel in the literature [2]. Likewise, the discrete preimage complexity function takes an important place in the theory: in the calculation of every preimage complexity function's induced partitioning, we first look at the distinct sigma matrices and their associated functions. Since H_d is our finest measure, each block of an induced partition $\mathcal{P}_{H_\sim}$ is thus some union of blocks of $\mathcal{P}_{H_d}$. From there, additional properties are added to consolidate blocks of $\mathcal{P}_H$ as required. See Example 6.1 for such an example.

Question: How many sigma matrices are there as a function of n, where $X = \{1, 2, 3, \ \ldots, \ n\}$? We will notate this sequence $\Sigma(n)$ and use the code in Appendix B to calculate the first few values

$$\Sigma(n) = \{1, 2, 5, 13, 35, 93, 260, \ \ldots\} \tag{12}$$

7. Directions for Future Research.

We have developed the theory of preimage complexity of endofunctions using the sigma matrix, a matrix of preimage set sizes at varying inverse depths. The data structures have been shown to be of interest in their own right -- having a natural order and illuminating characteristics of their generating functions. Likewise the preimage complexity functions are shown to be useful and flexible. In addition, we exhibit a novel sequence $\Sigma(n)$ corresponding to the number of equivalence classes of X^X as a function of the size of X and induced by equality of sigma matrices has been described.

There is much work to be done, and many questions to be asked. Below we list three possible areas of inquiry which would be useful in expanding the developed framework.

1. We would like to know more about the sequence $\Sigma(n)$ from Section 6. In particular, knowing if $\Sigma(n)$ is a super-sequence or sub-sequence of a known sequence would given insight into the structure of the iterated inverses of endofunctions or may connect this particular approach to other areas in the literature. Knowing if there a closed or recursive definition of $\Sigma(n)$ would be particularly interesting. Calculation of the sigma matrix of f is particularly efficient using our method in Section 2 where we prove that $\Sigma_f = \mathrm{ColSum}(F^{+})$. However our particular code, using libraries in the SageMath package of Python 3.X readily is overwhelmed by the combinatorial explosion as this requires the calculation of all sigma matrices on n^n functions, and the identification of duplicates under permutation of rows.

2. *Generalize the preimage structures, particularly the sigma matrix, to be useful in the analysis of non-functional relations on a finite set X.* Finite endofunctions $f : X \to X$ are a rich environment and provide a natural structure for the analysis of functional digraphs. However, many real-world systems are non-functional. An extension to more general relations on $X \times X$ would be a major step towards broadening the set of possible applications of the preimage approach. One approach to this is to adapt the preimage matrix F^{-} to contain multiple sets per entry by adapting the image matrix to be a matrix of sets, not

simply singleton entries. Consider the following example.

Example 7.1. Consider the relation $R = \{(a,\ b),\ (b,\ a),\ (b,\ c),\ (c,\ c)\}$ which fails to be functional as the image of b is $\{a,\ c\}$.

We may start by attempting to compute our data structures (the image, preimage and sigma matrices) as we did for functional relations. Starting with the image matrix, we find the rows of the sigma matrix and then construct the matrix itself as follows:

$$R^+[a,\ *] = [\{b\}, \{a,\ c\}, \{\{b\}, \{c\}\}],$$

$$R^+[b,\ *] = [\{a,\ c\}, \{\{b\}, \{c\}\}, \{\{a,\ c\}, \{c\}\}],$$

$$R^+[c,\ *] = [\{c\}, \{c\}, \{c\}]$$

$$R^+ = \begin{pmatrix} \{b\} & \{a,\ c\} & \{b\}, \{c\} \\ \{a,\ c\} & \{b\}, \{c\} & \{a,\ c\}, \{c\} \\ \{c\} & \{c\} & \{c\} \end{pmatrix}$$

Now, since each entry of F^+ when functional is a singleton, we adapt "element" to mean set for our relation R. So we can take the ColSum as usual except where the unique elements of the matrix are sets. For each entry X_i we calculate ColSum(R^+, $X_i \subset X$) and construct the "Relational Sigma Matrix"

$$\text{ColSum}(R^+,\ \{b\}) = [1,\ 1,\ 1],\ \ \text{ColSum}(R^+,\ \{c\}) = [1,\ 2,\ 3],\ \text{ColSum}(R^+,\ \{a,\ c\}) = [1,\ 1,\ 1]$$

$$\Sigma_R = \text{ColSum}(R^+) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

It should be noted here that the sum of each column of the sigma matrix is incremented by 1. Note that $|Im(b)| = 2$ and $\text{ColSum}(R^+,\ \{b\}) = [1,\ 1,\ 1]$. The aforementioned ColSum tells us that the split resulting in an additional set element happens once per column and thus so does the addition of 1 to the sum of the columns of the sigma matrix.

$$\sum_{i=1}^{n} \sigma_{(i,j)} = 1 + \sum_{i=1}^{n} \sigma_{(i,j-1)}$$

Such an alteration of the sigma matrix could then be incorporated as to examine the number of and multiplicity of non-functional splits per iteration and thus per row of the sigma matrix. With such an alteration of our main data structures it would be useful to know if similar insights into the associated digraphs can be constructed. Likewise we would like to know if there an ordering the set of these non-constant column-sum sigma matrices and if a similar measure of complexity can be developed.

3. *Define a more general composition poset on Σ_X.* Our composition poset of $(\Sigma_{X^X}, \leq)$ is particularly useful in that it does indeed generate a substrate upon which complexity measures with order can be built. In particular, as defined in Definition 3.12, the poset highlights an ordering with respect to injectivity/bijectivity to which the consolidation of values in the sigma matrices similarly belies. This is however a "first-order" or "first column" effect: it is constructed via single composition and corresponds only to information contained in the first column of the sigma matrix. In particular, as defined, the poset of Sigma Matrices has the property that if two sigma matrices are poset equivalent then their graphs have the same degree sequence. More succinctly, if two matrices are equivalent in the poset then their first columns are equal.

$$\Sigma_f \sim \Sigma_g \Longrightarrow \Sigma_f[\,*\,,1] = \Sigma_g[\,*\,,1]$$

Ideally, any poset structure on the sigma matrices should take into account more than just the first column of an n column matrix. We suggest that the maximum length path $P_{\max}(G)$ through a given function's graph should determine the number of columns required to specify this equivalence. Given $f,\ g: X \to X$, if $|P_{\max}(G(f))| > |P_{\max}(G(g))|$ and $f^m = f^l$ and $g^{m'} = g^{l'}$ where $l < m < n$ and $l' < m' < n$, then $min(m)$ should be the maximum and $min(m')$ the minimum number of columns of the sigma matrices considered in the determination of order in the poset of sigma matrices.

In fact we claim the following. Let D be a set of functions whose sigma matrices have the same first column I.e., for any $f,\ g \in D,\ \ \Sigma_f[\,*\,,1] = \Sigma_g[\,*\,,1]$. Then if α is the bijection with a single component (of size $n = |X|$) then the set $D \subseteq \{\alpha^1 \circ f,\ \alpha^2 \circ f,\ \ldots,\ \alpha^n \circ f\}$ for any choice of $f \in D$.

Said another way, given a function f with a graph of degree sequence d, by applying α we can reach any other function with a graph having the same degree sequence. Thus the poset structure we have imposed on our sigma matrices are only "first column sensitive". Using a poset structure which depends on all relevant depths of the inverse would be preferable.

References

1. Wolfram, S. (2002). A new kind of science (Vol. 5, p. 130). Champaign, IL: Wolfram media.

2. OEIS Foundation Inc. (2020), The On-Line Encyclopedia of Integer Sequences, http://oeis.org

Appendix A:
Notation and Conventions

$X^X = \{f : X \to X\}$ all endofunctions on X

$X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\} = \operatorname{dom}(f)$

$f^{-j}(y) := \left\{ x : f^j(x) = y \right\}$

$f^{-j}(A) := \bigcup_{x \in A} f^{-j}(x).$

$f^{-[1,n]}(x) := \left[f^{-1}(x),\ f^{-2}(x),\ \ldots,\ f^{-n}(x)\right].$

$f^{-j}[X] := \left[f^{-j}(x_1),\ f^{-j}(x_2),\ \ldots,\ f^{-j}(x_n)\right].$

$F^{-} := f^{-[1,n]}[X] \ : \ F^{-}[i, j] := f^{-j}(x_i)$

Σ_f the sigma matrix of f

$\Sigma_f[i, j] := |F^{-}[i, j]| = \left|f^{-j}(x_i)\right|$

$\Sigma_f[i, j] := \sigma_{(i,j)}$

$(\Sigma_X, \ \leq)$ composition poset of sigma matrices

Σ_{X^X} the set of all sigma matrices of X^X

$\Sigma_{\{a,b\}^{\{a,b\}}}$ set of all sigma matrices on functions $\{a, b\} \to \{a, b\}$,

$f^{[1,n]}(x) = [\, f(x), f^2(x), \ldots,\ f^{n-1}(x),\ f^n(x)\ ].$

$F^{+} := f^{[1,n]}[X] = \left[f^{[1,n]}(x_1), f^{[1,n]}(x_2),\ \ldots,\ f^{[1,n]}(x_n)\right] \ : \ F^{+}[i, j] := f^j(x_i)$

$F^{+}[i, j] := f^j(x_i)$

$M[i, \ *] := [M_{(i,1)},\ M_{(i,2)},\ \ldots,\ M_{(i,n)}]$ ith row of an $n \times n$ matrix M

$M[\, * \,, j] := [M_{(1,j)},\ M_{(1,j)},\ \ldots,\ M_{(1,j)}]$ jth column of an $n \times n$ matrix M

$F^{+}[x_i, \ *] = f^{[1,n]}(x_i)$

ColSum $(M \ : \ x)$ list counting the x' s in each column of the square matrix M

$M_{\text{cs}} = \text{ColSum}(M : X)$ matrix made up of counts for all elements in each column of M

$M_{\text{cs}}[i, j] = \text{ColSum}(M : x_i)\ [j]$ count of x_i' s in the jth column of an $n \times n$ matrix M

$\{\, * \,\}^{n \times n}$ an $n \times n$ matrix of singletons

P_{cs} procedure calculating the ColSum(M) from M

P_{+} procedure computing the image matrix F^{+} from f

$P_{\Sigma} : X^X \to \Sigma_{X^X}$

$P_\Sigma := P_{\text{cs}} \circ P_+$

$\alpha : X \text{ −> } X$ bijections on X

$c : X \text{ −> } X$ constant functions on X

$H : X^X \to S$ preimage complexity function, S finite set

$H := K \circ P_\Sigma$

$H_\sim$ an arbitrary preimage complexity function

$K : \Sigma_{X^X} \to S$

H_d the discrete preimage complexity function where $K_d(\Sigma) = \Sigma$

H_X the indiscrete preimage complexity function (see Example 5.3 for definition of K_X)

H_0 bijectivity preimage complexity measure.

$P \vdash n$ an integer partition of n

$P \vdash X^X$ partition of all functions $X \to X$

$\mathcal{P}_{H_d}, \mathcal{P}_{H_X}, \mathcal{P}_{H_0}$ the partitions of X^X induced by H_d, H_X and H_0

$[g]_{H_\sim}$ the partition element containing representative $g \in P_i \in \mathcal{P} \vdash X^X$ induced by $H_\sim$

$[f]_{=r} = \{f : H(f) = r\}$ when context makes the PCF in use clear.

$|\mathcal{P}_{H_\sim}(3)|$ the size of the partition of X^X induced by $H_\sim$ where $|X| = 3$

$\mathcal{P}_{H_\sim}(n)$ the sequence of sizes for partitions of X^X where for domain sizes $\{1, 2, 3 \ldots\}$

$\Sigma\,(n)$ sequence of the number of unique sigma matrices as a function of $\left|\text{dom}\,(f)\right| = n$

Appendix B
Calculation of Sequence $\Sigma(n)$

```
# Calculation of Novel Sequence Sigma(n)
import sagemath

def reverse_numeric(x, y):    #optionally used for sorting sigma matrices from nonzero rows to zeros, default is reverse of this
    return y - x

def sigSet(n):
    l=list(FiniteSetMaps(n))   # create the list of finite functions
    sigMat=set()             # prepare set of sigMats
    for f in l:              #build sigma matrices for each
        comps=[list(f)]      #comps is the auxiliary matrix of compositions of f
        u=f                  # initialize, first column
        for i in range(1,n):    # apply function to itself as many times as necessary
            v=[]
            for k in u:
                v.append(f[k])
            comps.append(v)    #comps is a list of lists at this point
            u=v
        m=matrix(comps)     # make it a matrix object
        m=m.transpose()       #take the transpose so rows become columns
        sm=matrix(n, n)       #prepare sigma matrix for f
        for i in range(n):
            for j in range(n):
                c=m.column(j)
                x=list(c).count(i)
                sm[i,j]=x         # by definition
        ssm=list(tuple(sm[i]) for i in range(n))
        ssm.sort()                              #sort the rows lexicographically to prepare setification
        essm=tuple(ssm)
        sigMat.add(essm)    # add to the collection of sigma matrices
    return sigMat
    print(sigMat)
q=sigSet(8); q; len(q)    # 8 can be changed to the desired index
#The Sage Notebook is based upon work supported by the National Science Foundation Grants
```

Vita

The author was raised in New Orleans, Louisiana. In 2020, he received his doctorate in mathematics at the University of New Orleans under Dr. Kenneth Holladay, a pupil of the famed mathematician and philosopher Gian Carlo Rota. He is a classically trained pianist and composer having studied under Dr. Raymond Gitz and Dr. Stephen Dankner respectively. Dr. Fournier-Eaton is an avid teacher: he has taught piano, music composition, secondary and collegiate math courses. He is a pilot and FAA Certified Flight Instructor. Also he is the author of several stock trading algorithms including the Trend Shift Indicator. He enjoys listening to music, studying tropical weather phenomena, and has a thirst for learning and creating. In 2018 Dr. Fournier-Eaton was married to Scott Fournier-Eaton. They live with their cat "Fizzy-Water"