Fall 12-18-2015

Towards a Theory of Recursive Function Complexity: Sigma Matrices and Inverse Complexity Measures

Bradford M. Fournier
University of New Orleans, bmfourni@uno.edu

Towards a Theory of Recursive Function Complexity:

Sigma Matrices and Inverse Complexity Measures

A Thesis

Submitted to the Graduate Faculty of the

University of New Orleans

in partial fulfillment of the

requirements for the degree of

Master of Science

In

Mathematics

By

Bradford Fournier

B.S. University of New Orleans, May 2013

December, 2015

For *C*, *H*, *F*, *O*, & *P*. As well for *J-B X*.

Acknowledgments

I am personally grateful to my adviser and mentor Professor Kenneth Holladay. His enthusiasm, knowledge and encouragement have been a sustaining force in my personal development and in that of this manuscript: I will always be grateful for the trust he placed in me - especially on the many days when my nascent ideas were hardly developed enough to be articulated. The patience, support, and kindness of others including Professor Ralph Saxton, Padi Fuster, Daniel Duarte, Aram Bingham and Patrick Roach should not go unremarked. I also thank Raymond Gitz - as much family as friend - who has been a critical support over many years. His infectious passion for music has given me a constant companion. Finally, I thank my family, especially my brother Christopher Eriksen. Our friendship and discussions have been a source of great joy for many years.

Contents

Abstract

This paper develops a data structure based on preimage sets of functions on a finite set. This structure, called the sigma matrix, is shown to be particularly well-suited for exploring the structural characteristics of recursive functions relevant to investigations of complexity. The matrix is easy to compute by hand, defined for any finite function, reflects intrinsic properties of its generating function, and the map taking functions to sigma matrices admits a simple polynomial-time algorithm . Finally, we develop a flexible measure of preimage complexity using the aforementioned matrix. This measure naturally partitions all functions on a finite set by characteristics inherent in each function's preimage structure.

Keywords: Sigma Matrix, Complexity, Preimage Structures, Algorithm, Combinatorics, Inverse Analysis.

1
Introduction

1.0 **A look ahead.** The literature is not short on definitions of complexity and no definition unites mathematicians, information theorists and natural scientists - and rightfully so. Any such definition should not be optimized with respect to an abstract standard of purity, but rather, be suited to the problem at hand. This paper serves to develop a useful framework in which to perform a complexity analysis of finite functions.

The choice is made to use preimage structures as the vantage from which to perform our analysis. In addition to being a natural way to attack problems of an information-theoretic or cryptographic bent, preimage structure analysis can be useful across many problem types. Suppose, for example, a chess player is required to start playing mid-game: the seasoned player will likely look for known lines along which the game is being played - to the game history - for context, direction and insight. We approach our complexity analysis analogously.

A data structure called the sigma matrix of f, denoted Σ_f, is herein developed. This structure possesses properties making it ideal for computing metrics on its associated function f. The sigma matrix is defined for all functions on a finite set, is simple to compute by hand (for small domains), and can detect when a domain element a cycle element in a function structure. Next, an exploration is made of the map $\vec{C}:\mathcal{F}_{\mathcal{X}} \to \Sigma_{\mathcal{F}_{\mathcal{X}}}$ which takes functions to sigma matrices.[1] This map may be linear-time implemented in the size of $\mathcal{X}$, and induces a partitioning of $\mathcal{F}_{\mathcal{X}}$ useful for exploring the structure of its domain. The product of $\vec{C}$, a sigma matrix, will be the vehicle by which we develop an adaptable measure of complexity.

1.1 **Underlying Objects and Notation.** We start by considering the immediate preimage set to an element y in the domain of a function f, i.e., $f^{-1}(y) = \{x : f^1(x) = y\}$. In addition to the constituents of the preimage set we may be interested in its order. For example, given two elements $y_1, y_2 \in \mathrm{Dom}(f)$, if the number of elements at inverse depth one are not equal, for example $\left| f^{-1}(y_1) \right| < \left| f^{-1}(y_2) \right|$, more time or space is required in specifying the elements of $\{x : f(x) = y_1\}$ than $\{x : f(x) = y_1\}$. Extending the inverse depth, we may consider $f^{-i}(y) = \{x : f^i(x) = y\}$ for $i \in \mathbb{N}$ and build a chain corresponding of $f^{-i}(y)$ as the inverse depth i increases to view a more complete genealogy of y under f:

$$\{x : f^1(x) = y\},\ \{x : f^1(x) \in \{x : f^1(x) = y\}\},$$

$$\{x : f^1(x) \in \{x : f^1(x) \in \{x : f^1(x) = y)\}\}\}.$$

The small scale objects under consideration will be standard: sets, lists, matrices and arrays. We will make use of structures built upon sets of preimage elements. Our two primary notational conventions are as follows: firstly, when the top-level object defined is a set, the argument is enclosed in parentheses, whereas if a list is referenced, brackets are used; secondly, if the inverse depth is allowed to range from 1 to the size of the domain, an arrow is used.

To illustrate the use of parenthesis indicating a set, consider the set of all elements which under j applications of f yield y, written $f^{-j}(y)$. When interested in the union of such sets for multiple domain elements, we write $f^{-j}(\mathcal{Y})$ to indicate $\cup_{y\in\mathcal{Y}} f^{-j}(y)$. A use of square brackets, as in $f^{-j}[\mathcal{X}]$, indicates the list of sets $f^{-j}(x)$ across the entire domain $\mathcal{X}$.

Finally, we employ an arrow to indicate a depth ordered list with depths ranging from 1 to j as in $\overset{\hookleftarrow}{f}^{j}[x]$. Compare a list of j-back preimage sets over $\mathcal{X}$

$$f^{-j}[\mathcal{X}] = \left[f^{-j}(x_1),\ f^{-j}(x_2),\ \ldots,\ f^{-j}(x_{|\mathcal{X}|})\right] \tag{1}$$

(2)

to a depth-ordered list of preimages to x to depth j.

$$\overset{\hookleftarrow}{f}^{j}[x] = \left[f^{-1}(x),\ f^{-2}(x),\ \ldots,\ f^{-j}(x)\right] \tag{3}$$

Next, we present the i^{th} row of a matrix - of rows - with each sub-list corresponding to a particular element's preimage depth ranging from 1 to $|\mathcal{X}|$.

$$\left[f^{-1}(x_i),\ f^{-2}(x_i),\ \ldots,\ f^{-|\mathcal{X}|}(x_i)\right] \in \overset{\leftarrow}{f}_{\mathcal{X}} := \overset{\hookleftarrow}{f}^{|\mathcal{X}|}[\mathcal{X}] \tag{4}$$

The reader should verify that $\overset{\hookleftarrow}{f}^{|\mathcal{X}|}[x_i]$ and $f^{-j}[\mathcal{X}]$ can be taken as the i^{th} and j^{th} row and column of $\overset{\leftarrow}{f}_{\mathcal{X}} := \overset{\hookleftarrow}{f}^{|\mathcal{X}|}[\mathcal{X}]$ respectively. Unless otherwise specified, it is our convention that $\mathcal{X}$ is domain set and $f_{\mathcal{X}}$ will mean "f with $\text{Dom}(f) = \mathcal{X}$" It is time for an example.

Example 1.1.1 Let $f_{\mathcal{X}} = \{(a, b), (b, a), (c, a), (d, b)\}$

To find $f^{-j}(x) : j = 3$ for each $x \in \{a, b, c, d\}$, we calculate the *three-back* inverse of each element of $\mathcal{X}$.

$$f^{-3}(\mathrm{a}) = \{b, c\}, \quad f^{-3}(b) = \{a, d\} \ , \quad f^{-3}(c) = \emptyset = f^{-3}(d)$$

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

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

For a single element b with inverse depths up to $|\mathcal{X}|$,

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

1.2 **The Preimage Matrix** It should be noted that most situations will be simplified by the fact that we will be most interested in examining $f^{-j}[x_i]$ where both j, i range over $[1, 2, \ldots, |\mathcal{X}|]$: namely a list of lists ranging over all elements of Dom(f) and all inverse depths from 1 to the size of the domain. The reader should verify that $\overleftarrow{f}^{|\mathcal{X}|}[x_i]$ and $f^{-j}[\mathcal{X}]$ can be taken as the i^{th} row and j^{th} column respectively in $\overleftarrow{f}^{|\mathcal{X}|}[\mathcal{X}]$. This list of lists and its notational conventions will be used often and thus given the simplified notation $\overleftarrow{f}_{\mathcal{X}}$. We will continue to use $\mathcal{X}$ for a set of domain elements and the notation $f(\mathcal{X})$ or $f_{\mathcal{X}}$ to mean "f with dom(f) = $\mathcal{X}$".

Example 1.2.1 *Calculating a Preimage Matrix.*

Let $f\{(a, b), (b, a), (c, a), (d, b)\}$, The preimage matrix is given by

$$\overleftarrow{f}^{|\mathcal{X}|}[\mathcal{X}] = \left[f^{-1}[\mathcal{X}], f^{-2}[\mathcal{X}], f^{-3}[\mathcal{X}], f^{-4}[\mathcal{X}]\right] \tag{5}$$

$$\overleftarrow{f}_{\mathcal{X}} = \begin{pmatrix} \{b, c\} & \{a, d\} & \{b, c\} & \{a, d\} \\ \{a, d\} & \{b, c\} & \{a, d\} & \{b, c\} \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix} .$$

Next, suppose the preimage matrices of α and β are different:

$$\begin{pmatrix} a & a & a \\ b & b & b \\ c & c & c \end{pmatrix} = \overleftarrow{\alpha}_{\mathcal{X}} \neq \overleftarrow{\beta}_{\mathcal{X}} = \begin{pmatrix} c & b & a \\ a & c & b \\ b & a & c \end{pmatrix}.$$

Despite these differences, all elements of the preimage matrices are singletons:

$$|\overleftarrow{\alpha}|_{\mathcal{X}} = |\overleftarrow{\beta}|_{\mathcal{X}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Both $|\overleftarrow{\alpha}|_{\mathcal{X}}$, $|\overleftarrow{\beta}|_{\mathcal{X}}$ are uniform in structure. This uniformity captures an important property: bijectivity. The value indicates that for each element y and any inverse depth j, $f^{-j}(y) = \{x : f^{j}(x) = y\} \in \overleftarrow{f}_{\mathcal{X}}$ is unique. Once a preimage matrix has been converted to one representing the size of its elements, there is no way to distinguish a three-component bijective function α from a single-component bijection , β.

As we set about to characterize the complexity of all functions on a finite domain, this loss of information will be an asset: there is a map from each function to an associated complexity class where multiple functions on the same domain may have preimage matrices with the same set sizes: resulting in a non-trivial partitioning of all functions on $\mathcal{X}$. This allows for an examination of structural similarities by examining a representative of each partition of $\mathcal{F}_{\mathcal{X}}$ under a given complexity measure.

2
The Sigma Matrix

2.1 **The Sigma Complexity Matrix.** With the desired characteristics of our complexity measure in mind, we now turn to developing our primary object for the analysis of preimage structures. We describe a row matrix where for each element in *Dom*(*f*) there is a row corresponding to the list of image elements under repeated application of *f*. That is, for each element $x \in Dom(f)$ there is a list $\vec{f}[x]$ of sets

$$\vec{f}[x] = [\, f(x),\, f^2(x),\, \ldots,\, f^{n-1}(x),\, f^n(x) \,] \quad s.t \quad n = |\mathcal{X}|.$$

Next, creating a matrix $\mathcal{M}$ where for each $x \in \mathrm{Dom}(f)$ the list $\vec{f}[x]$ is a row of $\mathcal{M}$. The resulting matrix of rows is the *forward-image* matrix of *f* over $\mathcal{X}$ notated $\vec{f}_{\mathcal{X}}$ (or just $\vec{f}$). An arbitrary element of the forward image matrix is given by,

$$\vec{f}_{ij} = f^j(x_i) \;:\; 1 \le i, j \le n = |\mathcal{X}|,$$

Recall that the preimage matrix $\overleftarrow{f}_{\mathcal{X}}$ is the list of lists where for each $x \in \mathrm{dom}(f)$ and each inverse depth *j* there is a row and column given by the following respectively:

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

$$\overleftarrow{f}^{j}[\mathcal{X}] = [f^{-j}(x_1),\, f^{-j}(x_2),\, \ldots,\, f^{-j}(x_n)] \;:\; \bigcup_{i=1}^{n} x_i = \mathcal{X}.$$

Thus, ij^{th} of $\overleftarrow{f}_{\mathcal{X}}$ is

$$\overleftarrow{f}_{ij} = f^{-j}(x_i) = \{x : f^j(x) = x_i\} \;:\; 1 \le i, j \le n.$$

The order of $\overleftarrow{f}_{ij}$ for all $1 \le i, j \le n$ will constitute the value of element *ij* in our primary data structure. We define it now.

Definition 2.1.1 *Let* $f : \mathcal{X} \to \mathcal{X} : |\mathcal{X}| = n$. *The sigma matrix of f* Σ_f *and its* ij^{th} *entry is then*

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

$$\text{Where } \sigma_i{}^j := \sigma_{ij} = \left|\overleftarrow{f}\right|^{-j}(x_i) = \#\left\{x_i : f^j(x) = x_i\right\}.$$

2.2 **Computation of Σ_f.** Preimage and sigma complexity matrices are simple enough. Computation of the sigma matrix, by a determination and count of preimage elements $\overleftarrow{f}_{ij}$, is not without its costs. For example, naively creating the three-by-three sigma matrix from the preimage elements of the constant function on {*a*, *b*, *c*} would require keeping track of twenty-two objects. Doing the same for {*a*, *b*, *c*, *d*} it requires a consideration of 52 objects - all to generate a matrix with only sixteen entries. Below is an example of such a brutish computation for the constant function on three elements with *a* the fixed point. Tracking preimages gives the following.

$$a \to \{a, b, c\} \to \{\ \{a, b, c\}, \{\}, \{\}\ \} \to \{\ \{a, b, c\}, \{\}, \{\}\ \}$$

$$b \to \{\} \to \{\} \to \{\}$$

$$c \to \{\} \to \{\} \to \{\}$$

To call this method cumbersome is generous. A procedure which constructs Σ_f in a well defined and reproducible way for any *f* is required. Before introducing this procedure, a convenient part of the calculation involved is codified in the following definition.

Definition 2.2.1 Let $\mathcal{L}$ be a list with i^{th}element $\mathcal{L}[i]$, and $\mathcal{M}$ be a matrix of $|\mathcal{L}|$ columns. Given some $x \in \mathcal{M}$, if $\mathcal{L}[i]$ counts the occurrences of *x* in the i^{th} column of $\mathcal{M}$, then $\mathcal{L}$ is a *column accumulator of x over* $\mathcal{M}$. We write Colsum($\mathcal{M}[x]$) to indicate the list formed by a column accumulation of *x* over $\mathcal{M}$. Considering the list of lists Colsum($\mathcal{M}[x]$) = $\mathcal{L}_x$ over all unique $x \in \mathcal{M}$, we write Colsum($\mathcal{M}[X]$) and call it the column accumulator matrix of $\mathcal{M}$.

Example 2.2.1 Calculation of a column accumulator matrix.

Suppose $\mathcal{M} = \begin{pmatrix} n & m & n \\ l & n & m \\ l & l & m \end{pmatrix}$.

To find the column accumulator matrix of $\mathcal{M}$, we consider the Colsum($M(x)$) for each of $\{l, m, n\}$.

Colsum($\mathcal{M}[l]$) = [2, 1, 0] , Colsum($\mathcal{M}[m]$) = [0, 1, 2] , Colsum($\mathcal{M}[n]$) = [1, 1, 1] .

Thus constituent rows formed from the above lists give the column accumulator of $\mathcal{M}$ over $\{l, m, n\}$ give

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

Example 2.2.2 $f = \{(i, l), (j, l), (k, m), (l, n), (m, l), (n, m)\}$

The reader should verify that the full column accumulator for $\vec{f}_{\mathcal{X}}$ is given by its *Colsum* as follows:

$$\vec{f}_{\mathcal{X}} = \begin{pmatrix} l & n & m & l & n & m \\ l & n & m & l & n & m \\ m & l & n & m & l & n \\ n & m & l & n & m & l \\ l & n & m & l & n & m \\ m & l & n & m & l & n \end{pmatrix} \implies \text{Colsum}(\vec{f}_{\mathcal{X}}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 3 & 2 & 1 \\ 2 & 1 & 3 & 2 & 1 & 3 \\ 1 & 3 & 2 & 1 & 3 & 2 \end{pmatrix}$$

Thus, calculation of Colsum($\vec{f}_{\mathcal{X}}$) is simply a matter of list creation and summing over elements of columns. Less expected is that there is simple way of constructing the sigma complexity matrix of f. The following result exploits this property of the image matrix while highlighting a relationship between a column accumulation over $\vec{f}_{\mathcal{X}}$ and the row elements of Σ_f.

Theorem 2.2.1: *If $\mathcal{C}$ is the column accumulator matrix over $\vec{f}_{\mathcal{X}}$, then $\mathcal{C}$ is also the sigma matrix Σ_f.*

Proof. First we show that $\dim(\vec{f}_{\mathcal{X}}) = \dim(\mathcal{C})$. If $\#\text{dom}(f) = n$ then $\dim(\vec{f}_{\mathcal{X}}) = (n \times n)$. Thus for each domain element of f there is a list $\mathcal{C}_x = \text{Colsum}\left(\vec{f}_{\mathcal{X}}[x]\right) \in \text{Colsum}(\vec{f}_{\mathcal{X}})$. Since $\vec{f}_{\mathcal{X}}$ has n columns, $\text{len}(\mathcal{C}_x) = n$ for all $x \in \text{dom}(f)$. Thus $\dim(\vec{f}_{\mathcal{X}}) = (n \times n) = \dim(\mathcal{C})$. Giving the following.

$$C_{ij} = \#\left\{x : x = x_i \in \left[f^j(x_1), f^j(x_2), \ \ldots, f^j(x_n)\right]\right\} = \#\left\{x : f^j(x) = x_i\right\} = |f|^{-j}(x_i)$$

Thus we have that $|f|^{-j}(x_i) = \sigma_i^j \in \Sigma_f$. Thus since $\dim(\vec{f}_{\mathcal{X}}) = \dim(\mathcal{C})$ and $C_{ij} = \sigma_i^j$ we conclude that $\mathcal{C} = \text{Colsum}(\vec{f}_{\mathcal{X}}) = \Sigma_f$.

■

We present pseudo-code for this process - a process admitting a polynomial-time algorithm.

```
𝒳 = Dom(f)
i = 1
n = size(𝒳)
'''
Let Colsum[f⃗_𝒳] = []
While i ≤ n:
    colsum(x_i) = []
        While j ≤ n:
            S = Σ_{j=1}^{n} γ(f⃗_ij) : γ(f⃗_ij) = { 1 , f_ij = x_i
                                                 { 0 , f_ij ≠ x_i
            append S to colsum(x_i)
            j = j + 1
    append colsum(x_i) to Colsum[f⃗_𝒳]
    i = i + 1
Σ_f = colsum[f⃗_𝒳]

return Σ_f
```

2.3 **Properties of the Sigma Matrix.** Having now defined the sigma complexity matrix and shown its construction from the image matrix, we move on to describing some nice properties of Σ_f. The following properties describe, directly or indirectly, the distribution of elements of the sigma complexity matrix. A property of $\Sigma_f$will be called local if it a statement about an individual element $\sigma_i^j \in \Sigma_f$. A property will be called sigma-global when its object is a column or row of Σ_f or the matrix itself.

Proposition 2.3.1: *All elements of the sigma complexity matrix are non-negative integers.*

Proof: The entries $\sigma_i^j \in \Sigma_f$ correspond to set sizes.

■

Proposition 2.3.2: *If x_i is a cycle element of $G(f)$ then $\sigma^j{}_i$ is non-zero for all j where $1 \leq j \leq n$.*

Proof: We have that $\sigma_i{}^j := |\overleftarrow{f}|^{-j}(x_i) = \#\{x_i : f^j(x) = x_i\}$. If $x \in C$ a cycle, x may also be the intersection of a tree $T \subset G(f)$ and C. Suppose first it is not an element of some tree entering C, then $|f|^{-1}(x) = 1$. However, if $x \in T \cap C$ then there exists y_1, y_2 such that when both elements y_1, y_2 are in the vertex set of $f$and without loss of generality y_1 is a cycle element, yet y_2 is not, then y_1, $y_2 \in f^{-1}(x)$ implies that the set $f^{-1}(x)$ is of size greater than one.
Finally if $x \in C$, there exists some $c \in C : c \in f^{-k}(x)$ for every $k \in \mathbb{N}$, so $|f|^{-j}(x) \geq 1$ for any j. This argument applies in the case of multi-component graphs.

■

Theorem 2.3.1: *If Σ_f is $n \times n$ then the sum of the elements in any column n .*

Proof: Since $\sigma_i^j = C_{ij} = \{x : f^j(x) = x_i\}$ and thus the j^{th} column of Σ_f has a sum given by *:*

$$\sum_{i=1}^{n} |\sigma_i^j| = \sum_{i=1}^{n} \# \{x : f^j(x) = x_i\} = \sum_{i=1}^{n} \# x_i \in c \in \vec{f}_X .$$

Since f is functional and onto, the set $\vec{f}^j[x_i]$ is always a singleton. Thus, for any i^{th}row of $\vec{f}_X$, $\# \vec{f}^j[x_i] = 1$. So,

$\sum_{i=1}^{n} \# x_i \in \vec{f}^j[X]$ must be the number of rows of $\vec{f}^j[X]$ which by must be the same as the rows of Σ_f, namely n.

■

Proposition 2.3.3: *If $x_i \in G(f)$ is not an element of a cycle then there exists some $k \leq n : \sigma_i{}^k = 0$.*

Proof: If x_i is not part of a cycle, it is part of a tree. Every recursive function f contains a cycle. If f contains only the trivial cycle, say $(c, c) \in E(G(f))$, then the distance from c to any element $x \neq c$ is less than or equal to $n - 1$. Thus the

distance of *a* leaf is less than or equal to $n - 1$. Lastly, for any leaf x_L, $f^{-1}(x_L) = \{\}$, thus for some $k \leq n$, $\sigma_i^k = 0$.

■

Theorem 2.3.2 : *If some element $\sigma_i^k \in \Sigma_f = 0$, then all remaining elements $\sigma_i^{k<j\leq n}$ of that row are also zero.*

Proof. Recall that σ_i^j is shorthand for $\left|f\right|^{-j}(x_i)$. Since $\left|f\right|^{-k}(x_i) = 0$, the associated set $f^{-k}(x_i) = \{x : f^k(x) = x_i\}$ is empty. Now, given an element $y \in f^{-j}(x_i)$, its preimage $f^{-1}(y)$ is an element of $f^{-(j+1)}(x_i) = \{x : f(x) \in \{f^j(x) = x_i\}\} = \{x : f(x) \in f^{-j}(x_i)\}$. In position k we have that the set $f^{-k}(x_i) = \emptyset$ and thus $f^{-(k+1)}(x_i) = \{x : f(x) \in f^{-k}(x_i)\} = \{x : f(x) \in \emptyset\}$. We also have that $f(x)$ exists for all $x \in \mathcal{X}$ and therefore $\{x : \vec{f}(x) \in \emptyset\}$ is itself empty. Since $\{x : f(x) \in \emptyset\} = \emptyset = f^{-(k+1)}(x_i)$ and so by definition of the sigma complexity matrix $\left|f\right|^{-(k+1)}(x_i) = \sigma_i^{k+1} = 0$.

■

Theorem: 2.3.3 : *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 for all $x \in \mathcal{X}$, $f(x) = c$ for some $c \in \mathcal{X}$. Let $c = x_k$, then $(x_k, f(x_k)) = (x_k, c) \in f$ and thus $x_k = c \in f^{-1}(c) = \mathcal{X}$ and $\left|f\right|^{-1}(c) = \left|\mathcal{X}\right| = n$. Inductively, if $c \in f^{-j}(c)$ then letting $D = f^{-j}(c)$, it must hold that $f^{-1}(c) \subset f^{-1}(D)$, and $c \in f^{-1}(c) \subset f^{-1}(D) = f^{-1}(f^{-j}(c))$ so $\left|f\right|^{-(j+1)} \geq n$. Now we show that $\left|f\right|^{-(j+1)} = n$. If $y \neq c \in f^{-j}(x_k)$, then $[f^{-1}(y) \subset f^{-1}(f^{-i}(c))] = \emptyset$ giving that $f^{-(j+1)}(c) = f^{-1}(c)$ Now since $\left|f\right|^{-1}(c) = \left|f\right|^{-(j+1)}(x_k) = n$ for any j, $\left|f\right|^{-j}(x_k) = \sigma_i^j = n$.

■

Theorem: 2.3.4: *The sum of all elements of Σ_f on n elements is n^2.*

Proof: Theorem 2.3.1 gives each column sum of *n*. The sigma matrix has been shown to have the number of columns as elements, *n*. It follows that the matrix has a total sum of n^2

■

Now that we have established some basic ‘niceness’ properties of the sigma matrix, we move to looking at correspondences between the sigma matrix and the graph of the associated function.

3

Graphic Thoughts

3.1 Function Digraphs. The utility of viewing a function as a set of ordered pairs is apparent when associating the function with its digraph. We use the standard notation $G(f)$ to indicate the graph of f. Many of our key results make use of the properties of function associated digraphs. This approach is appealing for several reasons: graph componants are easy to see, the notion of morphism - arrow as object - gives a visual structure to the underlying ordered pair object, and injectivity / surjectivity is surveyable for small domains suitably arranged. The aforementioned niceties correspond to procedures which are relevant in many complexity measurements: substructure detection , measuring (co)domain compression/consolidation, as well as branch counting in trees. Restricting ourselves to finite functions, we have a relation f where $\forall\, x \in \mathcal{X}\ \exists !\, y:\ x\, f\, y$. Then given any two ordered pairs in the set defining f, we have that if the second elements are different, the first elements must be different. Viewing $G(f)$, the graph of f, for each domain element there exists a unique directed edge - or arrow- such that the tail corresponds to the domain element and head to the image element. $\mathrm{Dom}(f) = \mathcal{X}$ corresponds to the vertices of $G(f)$ and is notated $V(G(f))$ or just $V(f)$.

The in-degree of a vertex v is the size of $\left\{x : f^{-1}(v) = v\right\}$, i.e., the number of arrows pointing to v. Of course f is a function and thus the number of arrows leaving any vertex element is always one. The set of all arrows, is notated as $E(G(f))$ or $E(f)$. Finally, the neighborhood of $y \in V(f)$ is a subset of $V(f)$ and made up of elements which are first-order image or preimage elements of y.

3.2 Graph Isomorphisms. Suppose that $f_{\mathcal{X}} = \{(a,\ b),\ (b,\ a),\ (c,\ a),\ (d,\ c)\}$ and $\pi_{\mathcal{X}} = \{(a,\ d),\ (b,\ b),\ (c,\ c),\ (d,\ a)\}$, then $f \circ \pi = \{(a,\ c),\ (b,\ a),\ (c,\ a),\ (d,\ b)\}$. It is evident that $G(f)$ and $G(f(\pi(x))$ are isomorphic - π, a permutation, serves to relabel the vertices set of $G(f)$. Correspondingly, there exists a permutation / relabeling of rows of Σ_f which produces the sigma matrix $\Sigma_{f\circ\pi}$.

The lexicographically row ordered matrices, given below, have differences which correspond only to differences in isomorphic graphs, thus

$$|\overleftarrow{f}|_{\mathcal{X}} = \left|f \overleftarrow{\circ} \pi\right| = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

That isomorphic graphs of two functions produce the same sigma matrix, and that the sigma matrix is the data structure upon which our exploration of complexity is based, require that any definition of complexity derived naturally conclude that *two functions with isomorphic graphs have the same sigma complexity.*

3.3 Information, Access, and Relative Bijectivity. As we develop our notion of complexity, flexibility is key. However, one reasonable notion of complexity across many domains is resistance to reversibility. Viewing a function f as a parametrized process, transformation, or operation, we examine a particular state the more preimage elements this state-element $f(y)$ has, the less certain we are about the state-element $f^{-1}(f(y))$. Under this view, one might decide that the bijectivity of a function is a relevant metric in a complexity determination. If the function is bijective, a unique preimage element can be specified. Thus, given any process which is bijective the state of the system n steps back from the current state can be known with certainty. In the case of the constant function however, there are $|\mathcal{X}|$ preimage elements for any target element (see below for a concrete example) . While the function $f = c$ has a very simple structure for $G(f) = G(c)$, the irreversibility of $f = c$ causes all ' information ' to be lost.

For example, suppose $g = \{(a, a), (b, a), (c, a)\}$. The union of all image elements under g is simply $\{a\}$, but the possible predecessors to a is all of $\mathcal{X}$. Compare this to $h = \{(a, b), (b, c), (c, a)\}$ where the set of image elements is all of $\mathcal{X}$ but each preimage set is a singleton. These results are important to our upcoming development of a preimage complexity measure.

4

The Complexity Measure

4.1 Definition and Properties. We previously developed a simple method for computing Σ_f - via the column accumulation over $\vec{f}$. This section animates the sigma matrix: we define a function $\mathcal{H}$ to flexibly capture the complexity of some finite f by way of preimage structures. To accomplish a comparison of complexitiy between two functions in $f_1, f_2 \in \mathcal{F}_\chi$, a notion un/equal with respect to $\mathcal{H}$ must be developed: a task accomplished by an equivalence relation $_{\mathcal{H}}\sim$ which induces a partitioning $\mathcal{P}_{_{\mathcal{H}}\sim}$ of $\mathcal{F}_\chi$ by $_{\mathcal{H}}\sim$. Thus the equivalence class of f (w.r.t. $_{\mathcal{H}}\sim$) is a set of functions defined by the following $[f]_\sim := \{g \in \mathcal{F} : g \sim f\}$. The ensuing discussion will develop these ideas with further precision.

Our complexity function $\mathcal{H} : f \to [f]_\sim$ will be determined only by properties of Σ_f. Thus, a natural property of $\mathcal{H}$ is that given $f_1, f_2 \in \mathcal{F}_\chi$ *s.t.* $f_1 \neq f_2$: $\Sigma_{f_1} = \Sigma_{f_2} \Longrightarrow \exists\ f_3 :\ f_1, f_2 \in [f_3]_\sim$ Two additional properties will inform our definition of the generalized complexity function $\mathcal{H}$. The contrasting structures of Σ_f for f_c , f_b a constant and bijection respectively will determine the possibility of some g such that f_c , $f_b \in [g]_\sim$. Also, the fact that for all permutations $\pi : \chi \to \chi,\ g = f(\pi(x)) \Longrightarrow\ \ \Sigma_f = \Sigma_g$ will inform our definition.

In summary, any generalized complexity function should be able to compare functions on a finite set by way of the sigma matrix, should incorporate the structural dichotomy between the constant and bijection, and not be sensitive to a relabeling of domain elements i.e., permutation of domain elements. Finally, there would be little use in a complexity function which could compare some characteristic of complexity of functions while being too coarse to ever put two functions in the same equivalence class. So, a final requirement is that the number of induced partitions under $\mathcal{H}$ is fewer than the order of the domain of the constituents of these partitions. We call this a sigma complexity function and define it as follows:

Definition 4.1.1 $\mathcal{H} : \mathcal{F}_\chi \longrightarrow \mathcal{A}$ is a *sigma complexity function* when for $f,\ g \in \mathcal{F}_\chi$ and finite $\mathcal{A}$:

1. $\Sigma_f = \Sigma_g \Longrightarrow \mathcal{H}(f) = \mathcal{H}(g)$
2. $\mathcal{H}(f) = \mathcal{H}(g) \iff \exists\ h \in \mathcal{F} : f,\ g \in [h]_\sim$ i.e., $f \sim_{\mathcal{H}} g$
3. If $\leq_{\mathcal{H}}$ is defined, let $\mathcal{A} = \mathbb{N}$ *s.t*

$\leq_{\mathcal{H}}(f, g) \iff \mathcal{H}(f) \leq \mathcal{H}(g)$

4. If f is bijective, and g constant then $\mathcal{H}(f) \neq \mathcal{H}(g)$.

It should be noted that $\mathcal{A}$ need not be ordered as $\mathcal{H}$ may simply partition $\mathcal{F}_{\mathcal{X}}$ rather than rank its functions. However, when $\leq$ is defined and $\mathcal{H}(\Sigma_f) = N_f \geq N_g = \mathcal{H}(\Sigma_g)$, we say that the sigma complexity of f is greater than or equal to that of g, etc.

Example 4.1.1 *The induced partitioning of $\mathcal{F}_{\mathcal{X}}$ by $\mathcal{H}$.*

The map $\vec{C}: \mathcal{F}_{\mathcal{X}} \to \Sigma_{\mathcal{F}_{\mathcal{X}}}$ is not surjective, by requiring $\Sigma_f = \Sigma_g \Longrightarrow \mathcal{H}(f) = \mathcal{H}(g)$, several functions $f \in \mathcal{F}_{\mathcal{X}}$ must have the same sigma matrix, and thus the same complexity under an unaltered definition of $\mathcal{H}$. For this example let $\mathcal{X} = \{a, b, c\}$. There are $|\mathcal{X}|^{|\mathcal{X}|} = 27$ functions of this set $\mathcal{F}_{\{a,b,c\}}$. Consider two functions f, g equal if $G(f) \cong G(g)$. Partitioning the functions this way yields seven different equivalence classes:

$$F_1 = \{\{(a, a), (b, a), (c, a)\}, \{(a, b), (b, b), (c, b)\}, \{(a, c), (b, c), (c, c)\}\}$$

$$F_2 = \{\{(a, a), (b, a), (c, b)\}, \{(a, b), (b, b), (c, a)\},$$
$$\{(a, b), (b, c), (c, c)\}, \{(a, a), (b, c), (c, a)\}, \{(a, c), (b, b), (c, b)\}, \{(a, c), (b, a), (c, c)\}\}$$

$$F_3 = \{\{(a, a), (b, b), (c, a)\}, \{(a, a), (b, b), (c, b)\}, \{(a, a), (b, a), (c, c)\},$$
$$\{(a, a), (b, c), (c, c)\}, \{(a, b), (b, b), (c, c)\}, \{(a, c), (b, b), (c, c)\}\}$$

$$F_4 = \{\{(a, b), (b, a), (c, a)\}, \{(a, b), (b, a), (c, b)\}, \{(a, b), (b, c), (c, b)\},$$
$$\{(a, c), (b, a), (c, a)\}, \{(a, c), (b, c), (c, a)\}, \{(a, c), (b, c), (c, b)\}\}$$

$$F_5 = \{\{(a, a), (b, c), (c, b)\}, \{(a, b), (b, a), (c, c)\}, \{(a, c), (b, b), (c, a)\}\}$$

$$F_6 = \{(a, b), (b, c), (c, a)\}, \{(a, c), (c, b), (b, a)\}$$

$$F_7 = \{\{(a, a), (b, b), (c, c)\}\}$$

It was established that two functions with isomorphic graph structures have the same sigma matrices. Thus we now take a representative f_i of each partition $\mathcal{F}_i$.

$$f_1 = \{\{(a, a), (b, a), (c, a)\}, f_2 = \{\{(a, a), (b, a), (c, b)\}$$

$$f_3 = \{\{(a, a), (b, b), (c, a)\}, f_4 = \{\{(a, b), (b, a), (c, a)\},$$

$$f_5 = \{\{(a, a), (b, c), (c, b)\}, f_6 = \{(a, b), (b, c), (c, a)\},$$
$$f_7 = \{\{(a, a), (b, b), (c, c)\}\}$$

This is not the complete picture. Recall that often two functions $f_{\mathcal{X}},\ g_{\mathcal{X}}$ exist such that $G(f) \ncong G(g)$, yet $\Sigma_f = \Sigma_g$. The sigma matricies are given for a representative of each class F:

$$\overleftarrow{f_1}| = \begin{pmatrix} 3 & 3 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad |\overleftarrow{f_2}| \begin{pmatrix} 2 & 3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$|\overleftarrow{f_3}| = \begin{pmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \qquad |\overleftarrow{f_4}| = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$|\overleftarrow{f_5}| = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad |\overleftarrow{f_6}| = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad |\overleftarrow{f_7}| = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Indeed, we find that the representatives of partitions 5, 6 and 7 all have the same sigma matricies.

Theroem 4.1.1 *: $\mathcal{H}(f_{\mathcal{X}}) = \mathcal{H}(f_{\pi(\mathcal{X})})$ for any $f_{\mathcal{X}}$ and any permutation $\pi : \mathcal{X} \to \mathcal{X}$.*

Proof: By graph isomorphism property, $G(f) \cong G(f \circ \pi) \implies \Sigma_f = \Sigma_{\pi(f)} \implies \mathcal{H}(f_{\mathcal{X}}) = \mathcal{H}(f_{\pi(\mathcal{X})})$.

■

4.2 Examples of Sigma Complexity Functions. It is time to introduce examples of a complexiy functions according to our example. The first, $\mathcal{K}_{\mathrm{Id}}$, while not imposing any extra constraints beyond those required by being a sigma complexity function, is non-trivial: it highlights ways in which for some $f,\ g$ on the same domain, $\Sigma_f = \Sigma_g$ while $f \neq g$.

Definition 4.2.1 *:* Let $\mathcal{K}_{\mathrm{Id}} : \mathcal{F}_{\mathcal{X}} \to \mathcal{A}$ be *induced sigma complexity function* with no extra constraints besides those imposed by the definition.

Example 4.2.1 : Let $\mathcal{X} = \{a,\ b,\ c,\ d\}, f = \{(a,\ b),\ (b,\ a),\ (c,\ a),\ (d,\ b)\}$,

Also let $g = \{(a, b), (b, b), (c, d), (d, d)\}$. Since $G(f)$ has one componant, and $G(g)$ has two, $G(f) \ncong G(g)$.

$$\vec{f} = \begin{pmatrix} b & a & b & a \\ a & b & a & b \\ a & b & a & b \\ b & a & b & a \end{pmatrix} \Longrightarrow \Sigma_f = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\vec{g} = \begin{pmatrix} b & b & b & b \\ b & b & b & b \\ d & d & d & d \\ d & d & d & d \end{pmatrix} \Longrightarrow \Sigma_g = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \Sigma_f$$

Thus, f, g are in the same partition of $\mathcal{F}_{\{a,b,c,d\}}$ under $\mathcal{K}_{\mathrm{Id}}$.

Definition 4.2.2 Let $\mathcal{K}_0$ be a sigma complexity function such that we call the *leaf counting measure,*

$$\mathcal{K}_0(f) := \sum_{i=1,j=1}^{n,n} \Delta\left(f_{\mathcal{X}}, \sigma_i^j\right) : \Delta\left(\sigma_i^j\right) = \begin{cases} 1, & \sigma_i^j = 0 \\ 0, & \sigma_i^j \neq 0 \end{cases}.$$

The function $\Delta\left(\sigma_{\mathrm{i}}^{\mathrm{j}}\right) = \begin{cases} 1, & \sigma_{\mathrm{i}}^{\mathrm{j}} = 0 \\ 0, & \sigma_{\mathrm{i}}^{\mathrm{j}} \neq 0 \end{cases}$ assigns a value of 1 if $(\sigma \in \Sigma_f) = 1$, otherwise it assigns 0 . Thus, summing over $\Delta\left(\sigma_i^j\right)$ for all $1 \leq i, j \leq n$, Δ produces a count of the number of zeros in Σ_f. That is to say that it counts all $y = f^{-j}(x_i) \in \overleftarrow{f} : f^{-1}\left(f^{-j}(x_i)\right) = \{\}$.

Theorem 4.2.1 *$\mathcal{K}_0(f_{\mathcal{X}}) = \mathcal{K}_0(f_{\pi(\mathcal{X})})$ for π a permutation of $\mathcal{X}$.*

Proof: Recall that to each element $x \in \mathcal{X}$ is associated a unique row of the complexity matrix Σ_f. Being a permutation, the bijectivity of $\pi : \mathcal{X} \to \mathcal{X}$ requires that $\{\pi(x) : x \in \mathcal{X}\} \cap \mathcal{X} = \mathcal{X}$; thus, for any row R in Σ_f it must also be in the matrix $\Sigma\, \overleftarrow{f}_{\pi(\mathcal{X})}$. Defining $\pi(x_i) := x_{\pi(i)}$, if $R = |\overleftarrow{f}|[x_i] \in \Sigma_f$, $R = |\overleftarrow{f}|\, x_{\pi(i)} \in \Sigma\, \overleftarrow{f}_{\pi(\mathcal{X})}$. Therefore, except for perhaps a permutation of rows, $\Sigma_f = \Sigma_{f_{\pi(\mathcal{X})}}$. So $\mathcal{K}_0\left(f_{\mathcal{X}}\right)$ is given by

$$\Delta\left(f_{\mathcal{X}},\ \sigma_i^j\right) = \sum_{i=1,j=1}^{n,n} \Delta\left(f_{\pi(\mathcal{X})},\ \sigma_i^j\right) = \mathcal{K}_0(f_{\pi(\mathcal{X})})$$

■

Theorem 4.2.2 *If $f,\ g \in \mathcal{F}_{\mathcal{X}}$ and if f is constant $\mathcal{K}_0(f) \geq \mathcal{K}_0(g)$.*

Proof: Since f is the constant function, Σ_f must have at least as many zeros as the sigma complexity matrix of any other function on $\mathcal{X}$, namely $n(n-1)$. If g is also a constant function then it holds that $\Sigma_f = \left[\Sigma \overleftarrow{g_{\mathcal{X}}}\right]$ and $\mathcal{K}_0(f) = \mathcal{K}_0(g)$ - a fact which follows from :

$$\sum_{i=1,j=1}^{n,n} \Delta(g_{\mathcal{X}},\ \sigma_i^j) = \sum_{i=1,j=1}^{n,n} \Delta(f_{\mathcal{X}},\ \sigma_i^j).$$

$$\mathcal{K}_0(g) = \sum_{i=1,j=1}^{n,n} \Delta(g_{\mathcal{X}},\ \sigma_i^j),$$

$$\sum_{i=1,j=1}^{n,n} \Delta(g_{\mathcal{X}},\ \sigma_i^j) \leq \sum_{i=1,j=1}^{n,n} \Delta(f_{\mathcal{X}},\ \sigma_i^j) \Longrightarrow \sum_{i=1,j=1}^{n,n} \Delta(f_{\mathcal{X}},\ \sigma_i^j) = n(n-1) = \mathcal{K}_0(f).$$

■

Theorem 4.2.3 *If $f,\ g \in \mathcal{F}_{\mathcal{X}}$ and if f is a bijection $\mathcal{K}_0(f) \leq \mathcal{K}_0(g)$.*

Proof: Since f is bijective, for all $x \in \mathcal{X}$ the size of the set $f^{-i}(x)$ is one and thus each entry of the associated sigma complexity matrix is likewise "1". Therefore there are no zeros in Σ_f, i.e., $\sum_{i=1,j=1}^{n,n} \Delta\left(f_{\mathcal{X}},\ \sigma_i^j\right) = 0 = \mathcal{K}_0(f)$. Now if g is also a bijection, then by the same logic.

$$\sum_{i=1,j=1}^{n,n} \Delta\left(g_{\mathcal{X}},\ \sigma_i^j\right) = 0 = \mathcal{K}_0(g)$$

and likewise if g is not a bijection then

$$\sum_{i=1,j=1}^{n,n} \Delta\left(g_{\chi},\ \sigma_i^j\right) \geq\ 1 = \mathcal{K}_0(g).$$

Thus, it is proven that if f is bijective then $\mathcal{K}_0(f) \leq \mathcal{K}_0(g)$.

■

Theorem 4.2.3 *$\mathcal{K}_0$ forms an is an equivalence relation on $\mathcal{F}$.*

Proof: Since $\mathcal{K}_0 : \mathcal{F} \to \mathbb{Z}^+$, together $\mathcal{K}(f_1) = N_1$ and $\mathcal{K}(f_2) = N_2$ give that $\mathcal{K}(f_1) = \mathcal{K}(f_2)$ *iff* $N_1 = N_2$. Letting $\sum_{i=1,j=1}^{n,n} \Delta\left(f_{\chi},\ \sigma_i^j\right) = N$, if $f_{\chi'} = f_{\chi}$ or $f_{\pi(\chi)}$, then $\Sigma_f = \left[\Sigma \overleftarrow{f'}_{\chi'}\right]$ up to isomorphism . Since $\Sigma_f = \Sigma \overleftarrow{f'}_{\chi'}$, the following identy holds:

$$\sum_{i=1,j=1}^{n,n} \Delta\left(f_{\chi},\ \sigma_i^j\right) = N = \sum_{i=1,j=1}^{n,n} \Delta\left(f'_{\chi},\ \sigma_i^j\right).$$

Therefore $\mathcal{K}_0(f') = \mathcal{K}_0(f_{\pi(\chi)}) = \mathcal{K}_0(f) = N$, hence $f \bar{\bar{\mathcal{K}}}_0 f$ for all f. Of course, by commutativity and trasitivitiy of $\mathbb{Z}^+$ under equality, $f_1 \bar{\bar{\mathcal{K}}}_0 f_2 \Longleftrightarrow f_2 \bar{\bar{\mathcal{K}}}_0 f_1$. Thus

$$\left(f_1 \bar{\bar{\mathcal{K}}}_0 f_2\right),\ \left(f_2 \bar{\bar{\mathcal{K}}}_0 f_3\right) \Longrightarrow \left(f_1 \bar{\bar{\mathcal{K}}}_0 f_3\right)$$

■

Vitae

My interests are varied and my course has been atypical - though, I wouldn't trade my winding path for one more direct even if I could. I am a certified flight instructor, pianist, composer, and have spent time living in a monastery. Currently living in New Orleans, I am proud to be a Ph.D student under Kenneth Holladay. I hope to develop the ideas in this thesis to further examine notions of preimage complexity. Wherever it leads, the future is bright.