LENGUAJES FORMALES Y ATRACTORES DE SIF

LUIS FERNANDO CELIS MANTILLA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2022

LENGUAJES FORMALES Y ATRACTORES DE SIF

LUIS FERNANDO CELIS MANTILLA

Trabajo de grado para optar al titulo de Matemático

Director
RAFAEL FERNANDO ISAACS GIRALDO
Msc. en Matemáticas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2022

Agradecimientos

Primero agradezco a mi familia y amigos por su constante apoyo y cariño durante mi formación académica. En especial a mis hermanos que siempre estuvieron para escuchar lo que decía. Quiero agradecer a cada profesor que participó en mi aprendizaje y también agradezco a la Universidad Industrial de Santander por ser el espacio que brindó durante mi estancia experiencias invaluables.

Tabla de Contenido

Introducción **3**

1 Preliminares **5**

1.1 Lenguajes formales 5

1.2 Geometría fractal 10

1.2.1 Sistemas iterados de funciones 12

1.2.2 Atractores de SIF 16

1.2.3 SIF con condensación 19

2 El laboratorio y el museo: aspectos experimentales **21**

2.1 Código en SageMath 22

2.2 Experimentación 25

3 Primera formalización **38**

4 Segunda formalización **41**

5 Discusión **45**

Referencias **46**

Lista de figuras

1 Autosemejanza en la naturaleza 3

1.1 Autómata que acepta el número impar de d's 9
1.2 Iteraciones del triangulo de Sierpinski 13
1.3 Primeras iteraciones del SIF 14
1.4 Función dirección en el conjunto de Cantor 15
1.5 Cuadrado relleno 16
1.6 Triángulo relleno 17
1.7 Recta rellena 18
1.8 Ramificación 18
1.9 Atractor de un SIF con condensación 20

2.1 Atractores que inspiraron el experimento 21
2.2 Transformación en el código 24
2.3 Automata de las palabras sin bb 25
2.4 Autómata de las palabras en que las d siempre siguen de c 27
2.5 Autómata que acepta palabras con exactamente 3 a's 28
2.6 Autómata de las palabras con un número impar de d's 29
2.7 Autómata de las palabras con una d 30
2.8 Autómata de las palabras de la forma $(a \vee b)^*(c \vee d)^*$ 31
2.9 Autómata de las palabras en que las b's y c's no siguen de b's o c's 32
2.10 Autómata de las palabras sin una letra consecutiva 33
2.11 Autómata de las palabras en que las b's que siguen de a's o d's que siguen de c's . 35
2.12 Autómata de las palabras en que las a's y b's no siguen de a's o b's 37

3.1 Árbol para el alfabeto binario 38

4.1 De fondo el atractor original y en colores el lenguaje $(a \vee b)^*(c \vee d)^*$ 42

Lista de tablas

2.1 Observaciones de las palabras sin bb 26
2.2 Observaciones de las palabras en que las d siempre siguen de c 27
2.3 Observaciones de las palabras que tienen exactamente 3 a's 28
2.4 Observaciones de las palabras con un número impar de d's 29
2.5 Observaciones de las palabras con una d 30
2.6 Observaciones de las palabras de la forma $(a \vee b)^*(c \vee d)^*$ 31
2.7 Observaciones de las palabras en que las b's y c's no siguen de b's o c's 32
2.8 Observaciones de las palabras sin una letra consecutiva 34
2.9 Observaciones de las palabras en que las b's que siguen de a's o d's que siguen de c's 36
2.10 Observaciones de las palabras en que las a's y b's no siguen de a's o b's 37

Resumen

TITULO: LENGUAJES FORMALES Y ATRACTORES DE SIF.[1]

AUTOR: LUIS FERNANDO CELIS MANTILLA[2]

PALABRAS CLAVE: AUTÓMATAS, SISTEMAS ITERADOS DE FUNCIONES, GEOMETRÍA FRACTAL.

DESCRIPCIÓN:

Los sistemas iterados de funciones (SIF) son el método clásico para generar fractales y para cada atractor de un SIF le corresponde un espacio de códigos asociados que es determinado por su número de funciones. Usando el alfabeto del espacio de códigos asociado se puede emplear la teoría de lenguajes formales para limitar el comportamiento del atractor de un SIF mediante el uso de un autómata finito determinista. En este trabajo de grado, se presenta desde un punto de vista experimental, tomando distintos SIF fijos que son afectados por una variedad de autómatas, cuyos atractores se exponen junto a algunas observaciones; para esto se programó un código que permita graficar dichos atractores y se concluye demostrando que estos atractores siguen viviendo en el espacio $\mathscr{H}(X)$.

[1] Trabajo de grado

[2] Facultad de Ciencias. Escuela de Matemáticas. Director: Rafael Fernando Isaacs Giraldo

Abstract

TITLE: FORMAL LANGUAGUES AND ATTRACTORS OF IFS [1]

AUTOR: LUIS FERNANDO CELIS MANTILLA[2]

KEYWORDS: AUTOMATA, ITERATED FUNCTION SYSTEMS, FRACTAL GEOMETRY.

DESCRIPTION:

Iterated function systems (IFS) are the classical method for generating fractals and for each attractor of an IFS corresponds to a space of associated codes that is determined by its number of functions. Using the associated codespace alphabet, formal language theory can be employed to limit the attractor behavior of an IFS by using a finite deterministic automaton. In this degree work, it is presented from an experimental point of view, taking different fixed IFSs that are affected by a variety of automata, whose attractors are exposed together with some observations; for this, a code was programmed allowing to graph said attractors and it is concluded by demonstrating that these attractors continue to live in space $\mathscr{H}(X)$.

[1] Bachelor thesis

[2] School of Mathematics. Faculty of Sciences. Director: Rafael Fernando Isaacs Giraldo

Introducción

Históricamente la geometría ha sido un pilar para la comprensión de nuestro entorno, medir era un concepto clave e intuitivo para sus aplicaciones, cuya generalización de dichos resultados fue el inicio de la geometría clásica; durante el siglo XX se establece la topología como una rama formal de estudio de las matemáticas, este campo logra ampliar la visión que se tenía sobre medir, para este siglo se heredaron varias inquietudes que la reciente topología podría intentar resolver, aparecen figuras extrañas como lo son la función de Weierstrass, la curva de Peano, el conjunto de Cantor. Hausdorff con su método para medir dimensiones halló que estas figuras poseían una dimensión distinta a la usual, en varias de ellas su dimensión era no entera, lo cual puso aún mas en duda lo que significaban estas propiedades, pero éste fue el comienzo para formalizar lo que significaban. Denotaremos la Dimensión de Hausdorff como D y la dimensión topológica como D_t, se consigue generalizar que la dimensión de estas figuras extrañas cumple lo siguiente:

$$D > D_t$$

Dada la irregularidad de esta propiedad Mandelbrot propone que a estas figuras se denominarán *fractales*, que proviene del latín *fractus* que significa fragmentar o separar , consolidó sus resultados afirmando que la geometría fractal es la indicada para describir la variedad de formas que es encuentran en la naturaleza, ya que una de las propiedades mas características de estas figuras es la autosimilitud o autosemejanza, que en palabras sencillas significa que se puede hallar copias de la figura en si misma, algunos ejemplos que se ven en la naturaleza pueden ser las nubes, las montañas o incluso el brócoli.

Figura 1

Foto del autor

A su vez a mediados del siglo XX con la computadora comenzando a establecerse como un instrumento nuevo para la ciencia, también se desarrolla las ciencias de la computación que se vio influenciada en sus inicios por los trabajos sobre la estructura lógica de las matemáticas hecho por Hilbert, Keene, Gödel y Turing entre otros. Además en benefició de la teoría de

la computación hecho por Turing, los estudios de Chosmky dieron lugar a la teoría de lenguajes formales que está ligada fuertemente a la teoría de autómatas que cuenta dentro de sus temas de estudio a los lenguajes regulares.

En este proyecto tomaremos estos hechos históricos anteriormente mencionados como punto de partida, buscaremos la manera de unir la teoría de lenguajes formales y la geometría fractal en un experimento sobre el comportamiento de las figuras fractales al limitarlas a un lenguaje, para esto en el capitulo 1 se proporcionará la teoría necesaria para comprender aspectos de la geometría fractal y los lenguajes formales que permitirán plantear un código de programación que se explicará en el capitulo 2, una vez lograda la exposición de dicho código se elaboran múltiples simulaciones que permitan abstraer relaciones y propiedades, con el fin de concretar dichos resultados se propone una formalización que será trabajada en el capitulo 3 y se da un vistazo en el capitulo 4 a otra propuesta de formalización. Para finalizar se incluirá un repertorio de inquietudes que la persona interesada en estas áreas de estudio pueda abordar.

1. Preliminares

Empezaremos presentando los conceptos claves que se necesitan para comprender el funcionamiento de un autómata finito determinista, el cual hace parte del estudio sobre lenguajes formales que son el pilar que sostiene la teoría de la computación, cuyo fruto nos favorece hoy en día. Para reforzar estos conceptos recomendamos ver (de Castro Korgi, 2004) y (Linz, 2006) que fueron las principales fuentes de consulta sobre el tema.

1.1. Lenguajes formales

Definición 1.1.1. Un **alfabeto** es un conjunto finito no vacío que denotaremos Σ, cuyos elementos se llaman **símbolos** o **letras**.

Definición 1.1.2. Una palabra o cadena sobre un alfabeto Σ es cualquier sucesión finita de elementos de Σ.

Ejemplo 1.1.3. Sea $\Sigma = \{a, b\}$ un alfabeto, conocido como *alfabeto binario*. Las palabras sobre este alfabeto son secuencias finitas de elementos compuestos por a's y b's , tales como:

- *aaaaa*;
- *aabababa*;
- *aabbaabb*;
- *bbbbbba*;

Definición 1.1.4. La palabra que no tiene símbolos se denomina **palabra vacía** y se denotará por λ.

Definición 1.1.5. El conjunto Σ^* denota el conjunto de todas las palabras que se forman con los símbolos del alfabeto Σ y se puede ver como:

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i,$$

donde Σ^i es el conjunto de palabras de longitud i. Además Σ^* se puede definir recursivamente así:

1. $\lambda \in \Sigma^*$;
2. Si $u \in \Sigma^*$ y $a \in \Sigma$ entonces $ua \in \Sigma^*$;
3. Las palabras de Σ^* se forman únicamente usando los items 1. y 2.

Ejemplo 1.1.6. Sea $\Sigma = \{a, b, c\}$, entonces

$$\Sigma^* = \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, abc, \dots\}.$$

Definición 1.1.7. (Longitud de palabra) Sea $\sigma \in \Sigma^*$ la longitud de una palabra es el número de símbolos de σ y se denota como $|\sigma|$. Se define de la siguiente manera:

$$|\sigma| = \begin{cases} 0 & \text{si } \sigma = \lambda; \\ n & \text{si } \sigma = a_1 a_2 a_3 \cdots a_n. \end{cases}$$

Ejemplo 1.1.8. En $\Sigma = \{a, b\}$ se tiene lo siguiente: $|aaaaaa| = |bbbbbb| = 6$, $|ab| = 2$ y $|baabaab| = 7$.

Definición 1.1.9. (Concatenación) Sean $\sigma, \beta \in \Sigma^*$, la **concatenación** de σ y β se denota por $\sigma \cdot \beta$ o $\sigma\beta$ y se define como:

1. $\sigma\lambda = \lambda\sigma = \sigma$.
2. Si $\sigma = a_1 a_2 a_3 \cdots a_n$ y $\beta = b_1 b_2 b_3 \cdots b_m$, entonces $\sigma\beta = a_1 a_2 a_3 \cdots a_n b_1 b_2 b_3 \cdots b_m$.

Proposición 1.1.10. *La concatenación es una operación que cumple la propiedad asociativa. Es decir si $\alpha, \beta, \omega \in \Sigma^*$, entonces*

$$(\alpha\beta)\omega = \alpha(\beta\omega).$$

Definición 1.1.11. (**Potencia de una palabra**) Dado $\omega \in \Sigma^*$ y $n \in \mathbb{N}$ tal que $n > 0$, se define:

$$\omega^0 = \lambda;$$
$$\omega^n = \underbrace{\omega\omega\omega\dots\omega}_{\text{n veces}}.$$

Definición 1.1.12. Sea L un subconjunto de Σ^*, L es llamado un **lenguaje** sobre el alfabeto Σ.

Nota: Como los lenguajes sobre Σ son subconjuntos de Σ^*, las operaciones entre conjuntos son validas para lenguajes es decir, para un par de lenguajes A, B cumplen $A \cup B$ la unión, $A \cap B$ la intersección, $A - B$ la diferencia y $A^c = \Sigma^* \setminus A$ el complemento.

Ejemplo 1.1.13. Para todo alfabeto Σ se tiene que:

$L = \emptyset$, es el lenguaje vacío;

$L = \Sigma^*$, es el lenguaje de todas las palabras sobre Σ;

$L = \{\lambda\}$, es el lenguaje de la palabra vacía..

Definición 1.1.14. (Concatenación de lenguajes) De manera análoga a la concatenación para símbolos podemos definir la concatenación para lenguajes, sean A y B lenguajes sobre Σ definimos:

$$AB = \{ab : a \in A \text{ y } b \in B\}.$$

Además, en general, no cumple la propiedad conmutativa es decir. $AB \neq BA$.

Ejemplo 1.1.15. Sea $\Sigma = \{a,b,c\}$, $A = \{a, ca, bb\}$ y $B = \{b,c\}$ entonces

$$AB = \{ab, ac, cab, cac, bbb, bbc\},$$

$$BA = \{ba, ca, bca, cca, bbb, cbb\}.$$

Se observa que $AB \neq BA$.

Propiedades de la concatenación de lenguajes. Sean A, B, C lenguajes sobre Σ. Entonces

1. $A \cdot \emptyset = \emptyset \cdot A = \emptyset$.
2. $A \cdot \{\lambda\} = \{\lambda\} \cdot A = A$.
3. La propiedad asociativa
$$A \cdot (B \cdot C) = (A \cdot B) \cdot C.$$
4. La propiedad distributiva con respecto a la unión,
$$A \cdot (B \cup C) = A \cdot B \cup A \cdot C;$$
$$(B \cup C) \cdot A = B \cdot A \cup C \cdot A.$$
5. Propiedad distributiva generalizada. Si $\{B_i\}_{i \in I}$ una familia de lenguajes sobre Σ, entonces
$$A \cdot \bigcup_{i \in I} B_i = \bigcup_{i \in I} (A \cdot B_i);$$
$$\left(\bigcup_{i \in I} B_i \right) \cdot A = \bigcup_{i \in I} (B_i \cdot A).$$

Definición 1.1.16. La **clausura** o **estrella de Kleene** de un lenguaje $A \subset \Sigma^*$ es el conjunto de todas las concatenaciones de palabras de A.

Definición 1.1.17. Un lenguaje es llamado **regular** si se pueden formar a partir de los lenguajes básicos $\emptyset$, λ, $\{a\}$, para cada $a \in \Sigma$, por medio de la unión, concatenación y la estrella de Kleene.

Definición 1.1.18. Para simplificar la manera en que se define un lenguaje regular se definen las **expresiones regulares** de la siguiente manera.

1. Las expresiones regulares básicas son:

   - $\emptyset$ es la expresión regular que representa al lenguaje $\emptyset$.
   - λ es la expresión regular que representa al lenguaje $\{\lambda\}$.
   - a es la expresión regular que representa al lenguaje $\{a\}$ con $a \in \Sigma$.

2. Si S y R son expresiones regulares de Σ entonces los siguientes resultados también son expresiones regulares.

   - La concatenación RS.
   - La unión $R \cup S$.
   - La estrella de Kleene R^*.

Entonces a un lenguaje regular le corresponde naturalmente una expresión regular.

Ejemplo 1.1.19. Sea $\Sigma = \{a, b, c\}$ las siguientes son expresiones regulares:

- $a(a \cup b \cup c)^*$ expresión regular de las palabras que inician con a.
- $(a \cup c)^*$ expresión regular de las palabras sin b's.
- $(b \cup c)^* a (b \cup c)^*$ expresión regular de las palabras que tienen exactamente una a.

Definición 1.1.20. Un autómata finito determinista M es una quíntupla $M = (Q, \Sigma, q_0, F, \delta)$ donde

1. $Q = \{q_0, q_1, \ldots, q_n\}$ es un conjunto finito de estados.
2. Σ un alfabeto finito y todas las palabras que procesa M son elementos de Σ^*.
3. $q_0 \in Q$ es llamado el estado inicial del autómata.
4. $F \subset Q$, con $F \neq \emptyset$ es el conjunto de estados finales o de aceptación.
5. δ es la función de transición definida como:

$$\begin{aligned} \delta : Q \times \Sigma &\to Q \\ (q_0, a) &\to q_i, \end{aligned}$$

con $0 \leq i \leq n$ diremos que el autómata termina de procesar una palabras w hasta que agote todos sus símbolos. Los autómatas finitos son representados por multigrafos y en el gráfico del autómata tomaremos el doble circulo como el estado de aceptación.

Definición 1.1.21. El conjunto de palabras aceptadas por un autómata M se denota por $L(M)$ y se define

$$L(M) = \{u \in \Sigma^* \mid M \text{ termina el procesamiento de } u \text{ en un estado } q_i \in F\}.$$

Ejemplo 1.1.22. Se definirá un autómata M que acepte solo las palabras que tengan un número impar de d's en un alfabeto de 4 símbolos, considere

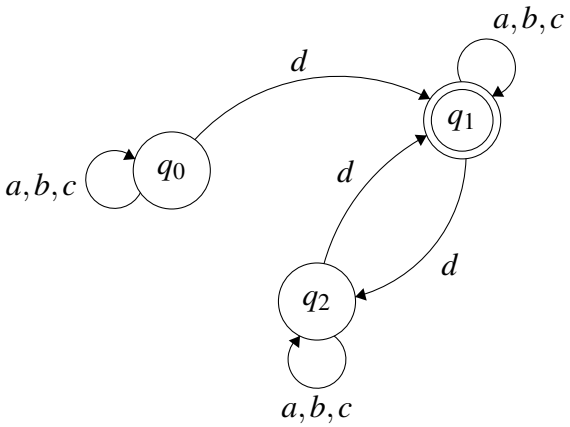
- $Q = \{q_0, q_1, q_2\}$ el conjunto de estados.
- $\Sigma = \{a, b, c, d\}$ el alfabeto.
- q_0 el estado inicial.
- $F = \{q_1\}$ el conjunto de estados de aceptación.
- y la función de transición δ estará dada por:

$$\begin{array}{llll} \delta(q_0,a) = q_0 & \delta(q_0,b) = q_0 & \delta(q_0,c) = q_0 & \delta(q_0,d) = q_1 \\ \delta(q_1,a) = q_1 & \delta(q_1,b) = q_1 & \delta(q_1,c) = q_1 & \delta(q_1,d) = q_2 \\ \delta(q_2,a) = q_2 & \delta(q_2,b) = q_2 & \delta(q_2,c) = q_2 & \delta(q_2,d) = q_1 \end{array}$$

El autómata finito que lo representa es:

Figura 1.1

Autómata que acepta el número impar de d's



Ahora considere las palabras *adcd* y *dadd* entonces su comportamiento a través de la

función δ es el siguiente.

$adcd$	$dadd$
$\delta(q_0,a)=q_0$	$\delta(q_0,d)=q_1$
$\delta(q_0,d)=q_1$	$\delta(q_1,a)=q_1$
$\delta(q_1,c)=q_1$	$\delta(q_1,d)=q_2$
$\delta(q_1,d)=q_2$	$\delta(q_2,d)=q_1$
$q_2 \notin F$	$q_1 \in F$
No es una palabra aceptada	Es una palabra aceptada

Teorema 1.1.23. ***(Teorema de Kleene)****. Un lenguaje es regular si, y solo si, es aceptado por un autómata finito.*

Demostración. Una prueba del teorema se puede encontrar en (de Castro Korgi, 2004, 49). □

Lema 1.1.24. ***(Lema del bombeo)****. Sea L un lenguaje regular, entonces existe un $n \in \mathbb{N}$ que llamaremos constante de bombeo, tal que para cualquier $w \in L$ con $|w| \geq n$ satisface que w se puede descomponer en 3 palabras $w = xyz$ tales que:*

1. $|xy| \leq n$
2. $y \neq \lambda$
3. *para todo* $i \geq 0$ *se tiene* $xy^iz \in L$

El lema del bombeo es una propiedad de los lenguajes regulares que está ligado con el principio de palomar y es comúnmente usado para mostrar que un lenguaje no es regular.

Ejemplo 1.1.25. En $\Sigma = \{a,b\}$ veamos que $L = \{a^nb^n : n \geq 0\}$ no es un lenguaje regular.

Supongamos que L es un lenguaje regular entonces existe $p > 1$ tal que para una palabra $w = a^pb^p \in L$ cumple que $|w| \geq p$, luego se puede descomponer $w = xyz$ tal que $|xy| \leq p$ y $y \neq \lambda$, pero esto significa que xy es una palabra de a's y y tiene al menos una a, es decir $x = a^k$ y $y = a^q$ tal que $k+q = p$ lo cual satisface las condiciones 1. y 2. del lema, luego $w = a^ka^qb^p$ ahora para $xy^iz \in L$ pero considere en particular $i = 2$ entonces $xy^2z = a^ka^{2q}b^p$ pero esto es contradicción ya que $k+2q \neq p$, por lo tanto no es un lenguaje regular.

1.2. Geometría fractal

En topología los espacios métricos fueron ampliamente estudiados ya que caracterizaban la noción de distancia, este concepto fue importante para el desarrollo de la geometría fractal como se ve en el problema medir la costa de Gran Bretaña, recomendamos ver (Barnsley, 2014) y

(Sabogal y Arenas, 2011) para reforzar estos conceptos, además para entrar en detalle sobre el espacio de los códigos se sugiere ver (Édgar, 1994).

Definición 1.2.1. (**Espacio de códigos**) Sea $\Sigma^{\mathbb{N}}$ el conjunto de sucesiones de un alfabeto $\Sigma = \{1,2,3,\ldots,N\}$ y definamos la métrica d_c como:

$$d_c(\alpha,\beta) = \sum_{i=1}^{\infty} \frac{|\alpha_i - \beta_i|}{(N+1)^i}$$

para todo $\alpha = \alpha_1\alpha_2\ldots$, $\beta = \beta_1,\beta_2\cdots \in \Sigma^{\mathbb{N}}$. La métrica d_c forma un espacio métrico en $\Sigma^{\mathbb{N}}$.

Definición 1.2.2. (**Espacio métrico completo**) Dado un espacio métrico (X,d) diremos que es completo, si toda sucesión de Cauchy en X converge en X.

Proposición 1.2.3. *El espacio de códigos $(\Sigma^{\mathbb{N}},d_c)$ cumple las siguientes propiedades.*

- *Es un espacio métrico completo;*
- *Es un espacio métrico compacto;*
- *Es un espacio métrico totalmente disconexo;*
- *Es homeomorfo al espacio de Cantor.*

La demostración de lo anterior se encuentra en (Édgar, 1994, pag 39,30,31,32).

Definición 1.2.4. Dada una función $f : X \to X$ en un espacio métrico (X,d) es llamada contracción si existe una constante $s \in \mathbb{R}$, $0 \leq s < 1$ tal que $\forall x,y \in X$ cumpla:

$$d(f(x),f(y)) \leq s \cdot d(x,y).$$

Definición 1.2.5. Sea (X,d) un espacio métrico. Llamaremos $\mathscr{H}(X)$ a la familia de todos los subconjuntos compactos no vacíos de X, es decir:

$$\mathscr{H}(X) := \{K \subset X \ : \ K \text{ es compacto, } K \neq \emptyset\}.$$

Definición 1.2.6. (**Distancia de un punto a un compacto**). Sea (X,d) un espacio métrico, $a \in X$ y $K \in \mathscr{H}(X)$ definimos

$$\hat{d}(a,K) := \text{mín}\{d(a,x) \mid x \in K\}.$$

Definición 1.2.7. (**Distancia entre compactos**). Sean (X,d) espacio métrico y $A,B \in \mathscr{H}(X)$ definimos:

$$\begin{aligned}\tilde{d}(A,B) &= \text{máx}\{\hat{d}(a,B) \mid a \in A\} \\ &= \text{máx}\{\text{mín}\{d(a,b) \mid b \in B\} \mid a \in A\}.\end{aligned}$$

Definición 1.2.8. (**Métrica de Hausdorff**). Sean $A,B \in \mathscr{H}(X)$ se define

$$h(A,B) := \text{máx}\{\tilde{d}(A,B),\tilde{d}(B,A)\}.$$

$(\mathscr{H}(X),h)$ es un espacio métrico y lo llamaremos el espacio donde viven los fractales.

Definición 1.2.9. Sean (X,d) un espacio métrico, $A \in \mathscr{H}(X)$ y $\varepsilon > 0$ se define la **nube** de centro A y radio ε, denotada como $N(A;\varepsilon)$ o $N_\varepsilon(A)$ y se define

$$N_\varepsilon(A) := \{x \in X \mid \hat{d}(x,A) < \varepsilon\}.$$

$$N_\varepsilon(A) = \{x \in X \mid d(x,a) < \varepsilon \text{ para algún } a \in A\}.$$

Con esta definición podemos reescribir la distancia de Hausdorff como :

$$D_h(A,B) = \text{ínf}\{\varepsilon > 0 \ : \ B \subset N_\varepsilon(A) \wedge A \subset N_\varepsilon(B)\}.$$

Lema 1.2.10. *Sean $A,B \in \mathscr{H}(X)$ y $\varepsilon > 0$. Entonces*

$$h(A,B) < \varepsilon \text{ si y sólo si } B \subset N_\varepsilon(A) \text{ y } A \subset N_\varepsilon(B).$$

1.2.1. Sistemas iterados de funciones

Definición 1.2.11. Sea X un espacio métrico completo, llamaremos **sistema iterado de funciones** brevemente SIF a la estructura $\{X; f_1, f_2, \ldots, f_N\}$ donde cada $f_i : X \to X$ con $i = \{1,2,3,\ldots,N\}$, es una contracción en X.

Definición 1.2.12. Dada $f : X \to X$ una aplicación en un espacio métrico. Las iteraciones hacía adelante de f son las aplicaciones $f^{\circ n} : X \to X$ definidas como $f^{\circ 0}(x) = x$, $f^{\circ 1}(x) = f(x)$, $f^{\circ n+1}(x) = f \circ f^{\circ n}(x) = f(f^{\circ n}(x))$ para $n = 0,1,2,\ldots$. Si f es invertible entonces las iteraciones hacía atrás de f son las aplicaciones $f^{\circ(-m)} : X \to X$ definidas como $f^{\circ(-1)}(x) = f^{-1}(x)$, $f^{\circ(-m)}(x) = (f^{\circ m})^{-1}(x)$ para $m = 1,2,3,\ldots$.

Teorema 1.2.13. *Dado un SIF $\{X; f_1, f_2, \ldots, f_N\}$ se define*

$$\begin{array}{rcl} F : \mathscr{H}(X) & \to & \mathscr{H}(X) \\ K & \to & F(K) =: \displaystyle\bigcup_{i=1}^{N} f_i(K) \end{array}.$$

Entonces existe un único $\mathscr{A} \in \mathscr{H}(X)$ tal que

$$F(\mathscr{A}) = \mathscr{A}.$$

Además, para cualquiera $K \in \mathscr{H}(X)$ se tiene que

$$\lim_{n\to\infty} F^{\circ n}(K) = \mathscr{A}.$$

La demostración de este resultado está basado en el teorema del punto fijo de Banach ya que F es una contracción y el ultimo resultado nos dice que tomando cualquier compacto su limite converge a $\mathscr{A}$.

Definición 1.2.14. El punto fijo $A \in \mathscr{H}(X)$ mencionado en el teorema anterior es llamado ***atractor*** del SIF.

Ejemplo 1.2.15. Considere el SIF $\{\mathbb{R}^2, f_1, f_2, f_3\}$ donde

$$f_1(x,y) = \frac{1}{2}(x,y) \tag{1.1}$$

$$f_2(x,y) = \begin{bmatrix} -\frac{1}{4} & \frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{3}{4} \\ \frac{\sqrt{3}}{4} \end{bmatrix} \tag{1.2}$$

$$f_3(x,y) = \begin{bmatrix} -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{3}{4} \\ \frac{\sqrt{3}}{4} \end{bmatrix} \tag{1.3}$$

Donde su atractor es conocido como el Triangulo de Sierpinski.

Figura 1.2

Iteraciones del triangulo de Sierpinski

Ejemplo 1.2.16. Considere el SIF $\{\mathbb{R}^2, f_1, f_2\}$ donde

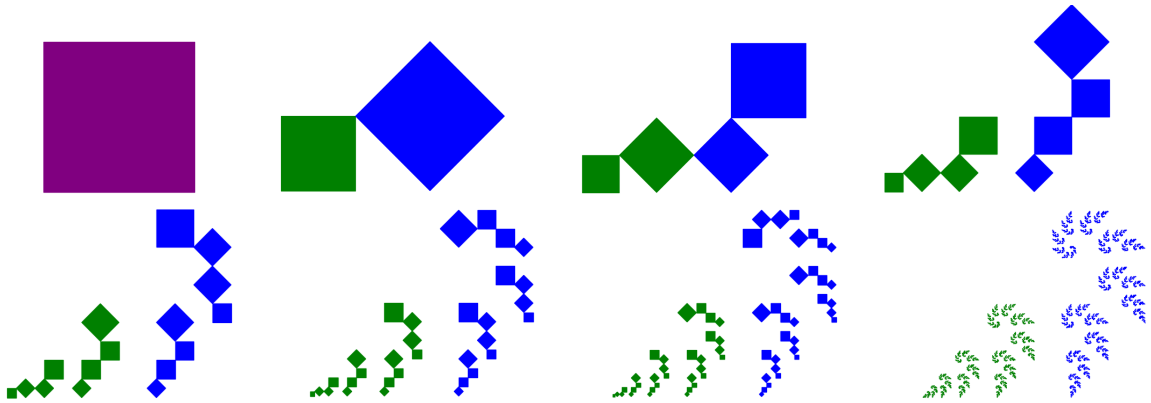
$$f_1(x,y) = \frac{1}{2}(x,y) \tag{1.4}$$

$$f_2(x,y) = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1.5}$$

Las iteraciones hasta llegar a su atractor se comportan de la siguiente manera:

Figura 1.3

Primeras iteraciones del SIF



Definición 1.2.17. Dado un SIF $\{X, f_1, f_2, \ldots, f_N\}$, se define el **espacio de códigos asociado** como el espacio métrico $(\Sigma^{\mathbb{N}}, d_c)$ definido en $\Sigma = \{1, 2, \ldots, N\}$.

Lema 1.2.18. *Dado un SIF $\{X, f_1, f_2, \ldots f_N\}$ con factor de contracción s, siendo $s = \max\{s_1, s_2, \ldots, s_n\}$ y cada s_i es el factor de contracción de cada f_i y $\Sigma^{\mathbb{N}}$ el espacio de códigos asociado al SIF. Para cada $\alpha \in \Sigma^{\mathbb{N}}, n \in \mathbb{N}$, y $x \in X$, se define*

$$\phi(\alpha, n, x) = f_{\alpha_1} \circ f_{\alpha_2} \circ \cdots \circ f_{\alpha_n}(x).$$

Sea $K \in \mathscr{H}(X)$. Entonces existe una constante D tal que:

$$d(\phi(\alpha, m, x_1), \phi(\alpha, n, x_2)) \leq D s^{\min\{m.n\}}$$

$\forall \alpha \in \Sigma^{\mathbb{N}}, \forall m, n \in \mathbb{N}$ y $\forall x_1, x_2 \in K$.

Demostración. Ver (Sabogal y Arenas, 2011, Lema 4.3.4). □

Teorema 1.2.19. *Dado un SIF $\{X, f_1, f_2, \ldots, f_N\}$, $\mathscr{A}$ su atractor y $\Sigma^{\mathbb{N}}$ el espacio de códigos asociado al SIF. Para cada $\alpha \in \Sigma^{\mathbb{N}}, n \in \mathbb{N}$, y $x \in X$, se define*

$$\phi(\alpha, n, x) = f_{\alpha_1} \circ f_{\alpha_2} \circ \cdots f_{\alpha_n}(x)$$

entonces

$$\varphi(\alpha) = \lim_{n \to \infty} \phi(\alpha, n, x)$$

Además $\varphi(\alpha)$ siempre existe, pertenece a $\mathscr{A}$, es independiente de x y la función

$$\begin{array}{rcl} \varphi : \Sigma^{\mathbb{N}} & \to & \mathscr{A} \\ \alpha & \to & \varphi(\alpha) \end{array}$$

es continua y sobre.

Demostración. Ver (Sabogal y Arenas, 2011, Teorema 4.3.5). □

Definición 1.2.20. (**Función dirección**). Sea $\{X, f_1, f_2, \ldots, f_N\}$ un SIF y $\Sigma^{\mathbb{N}}$ el espacio de códigos asociado y $\varphi : \Sigma^{\mathbb{N}} \to A$ la función definida en el teorema anterior, sea $a \in A$ llamaremos dirección o código de a a cualquier elemento del conjunto

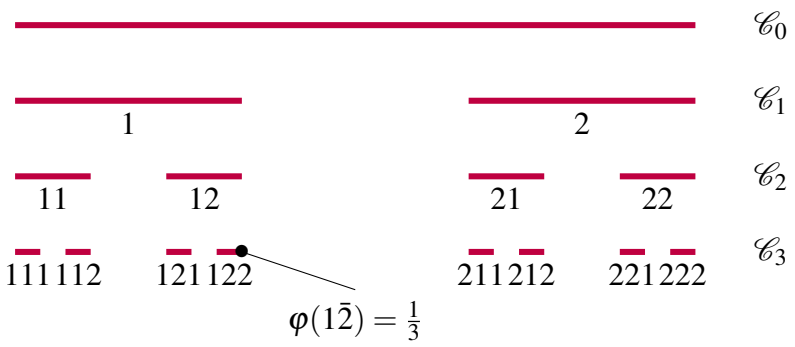
$$\varphi^{-1}(a) := \{\alpha \in \Sigma^{\mathbb{N}} : \varphi(\alpha) = a\}$$

El conjunto $\varphi^{-1}(a)$ lo llamaremos el conjunto de direcciones de a y la función φ la llamaremos **función dirección** o **función direccionamiento** del atractor del SIF.

Ejemplo 1.2.21. Sea el SIF $\{[0,1]; f_1, f_2\}$, donde $f_1(x) = \frac{1}{3}x$, $f_2(x) = \frac{1}{3}x + \frac{2}{3}$, el atractor es el conjunto ternario de Cantor $\mathscr{C}$ y su espacio de códigos es sobre el alfabeto $\Sigma = \{1,2\}$.

Figura 1.4

Función dirección en el conjunto de Cantor



Lema 1.2.22. *Sea A el atractor de un SIF $\{X, f_1, f_2, \ldots, f_N\}$ y $\varphi : \Sigma^{\mathbb{N}} \to A$ la función direccionamiento del SIF. Entonces para todo $\alpha \in \Sigma^{\mathbb{N}}$ se cumple*

$$\varphi(\alpha_1\alpha_2\alpha_3\ldots) = f_{\alpha_1}(\varphi(\alpha_2\alpha_3\alpha_4\ldots))$$

A continuación usaremos esta base teórica para describir el experimento.

1.2.2. Atractores de SIF

En esta sección presentaremos una variedad de SIF's junto con su atractor que se usarán para experimentar.

Cuadrado relleno. Para $(x,y) \in \mathbb{R}^2$ se define el siguiente SIF:

$$f_1(x,y) = \frac{1}{2}(x,y) \tag{1.6}$$

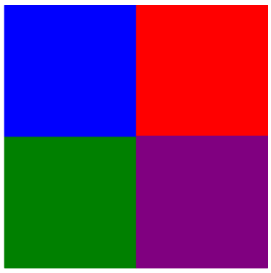
$$f_2(x,y) = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.7}$$

$$f_3(x,y) = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1.8}$$

$$f_4(x,y) = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{1.9}$$

Figura 1.5

Cuadrado relleno



Triangulo relleno. Para $(x,y) \in \mathbb{R}^2$ se define el siguiente SIF:

$$f_1(x,y) = \frac{1}{2}(x,y) \tag{1.10}$$

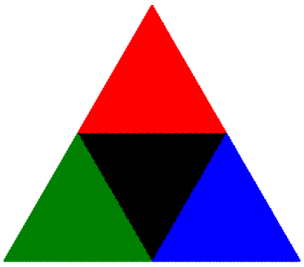
$$f_2(x,y) = \begin{bmatrix} -\frac{1}{4} & \frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{3}{4} \\ \frac{\sqrt{3}}{4} \end{bmatrix} \tag{1.11}$$

$$f_3(x,y) = \begin{bmatrix} -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{3}{4} \\ \frac{\sqrt{3}}{4} \end{bmatrix} \tag{1.12}$$

$$f_4(x,y) = \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \frac{3}{4} \\ \frac{\sqrt{3}}{4} \end{bmatrix} \tag{1.13}$$

Figura 1.6

Triángulo relleno



Recta rellena. Para un $x \in \mathbb{R}$ se define el siguiente SIF:

$$f_1(x) = \frac{1}{3}x \tag{1.14}$$

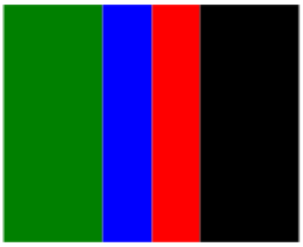
$$f_2(x) = \frac{1}{6}x + \frac{1}{3} \tag{1.15}$$

$$f_3(x) = \frac{1}{6}x + \frac{1}{2} \tag{1.16}$$

$$f_4(x) = \frac{1}{3}x + \frac{2}{3} \tag{1.17}$$

Figura 1.7

Recta rellena



Ramificación. Tomado de (Prusinkiewicz y Hammel, 1991) y se define para $(x,y) \in \mathbb{R}^2$ se define el siguiente SIF:

$$f_1(x,y) = \frac{1}{2}(x,y) \tag{1.18}$$

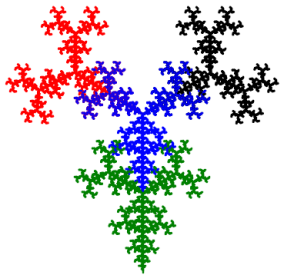
$$f_2(x,y) = \frac{1}{2}(x,y) + \begin{bmatrix} 0 \\ \frac{1}{2} \end{bmatrix} \tag{1.19}$$

$$f_3(x,y) = \begin{bmatrix} \frac{\cos(45)}{2} & -\frac{\text{sen}(45)}{2} \\ \frac{\text{sen}(45)}{2} & \frac{\cos(45)}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.20}$$

$$f_4(x,y) = \begin{bmatrix} \frac{\cos(-45)}{2} & -\frac{\text{sen}(-45)}{2} \\ \frac{\text{sen}(-45)}{2} & \frac{\cos(-45)}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.21}$$

Figura 1.8

Ramificación



1.2.3. SIF con condensación

Definición 1.2.23. Sean (X,d) un espacio métrico y $C \in \mathscr{H}(X)$.Definimos $w_0 : \mathscr{H}(X) \longrightarrow \mathscr{H}(X)$ por $w_0(K) := C, \forall K \in \mathscr{H}(X)$. Diremos que w_0 es una transformación de condensación, y C se llama el conjunto de condensación asociado.

Nota: Observe que $w_0 : \mathscr{H}(X) \to \mathscr{H}(X)$ viene siendo una función constante en $(\mathscr{H}(X), d_H)$ con factor de contracción igual a cero, cuyo punto fijo es el conjunto de condensación C.

Definición 1.2.24. Sean $\{X; w_1, w_2, \ldots, w_N\}$ un SIF y $w_0 : \mathscr{H}(X) \longrightarrow \mathscr{H}(X)$ una transformación de condensación. Entonces $\{X; w_0, w_1, w_2, \ldots, w_N\}$ se llama un SIF con condensación.

Teorema 1.2.25. *Sea $\{X; w_0 w_1, w_2, \ldots, w_N\}$ un SIF con condensación. Entonces la transformación $W : \mathscr{H}(X) \to \mathscr{H}(X)$ definida por*

$$W(K) := \bigcup_{i=0}^{N} w_i(K), \quad K \in \mathscr{H}(X)$$

tiene un único punto fijo, es decir, existe un único $\mathscr{A} \in \mathscr{H}(X)$ tal que:

$$W(\mathscr{A}) = \mathscr{A} = \bigcup_{i=0}^{N} w_i(\mathscr{A})$$

Además, para cualquier $K \in \mathscr{H}(X)$ se tiene que:

$$\lim_{n\to\infty} W^{\circ n}(K) = \mathscr{A}.$$

El conjunto $\mathscr{A}$ que cumple lo anterior se llama el atractor del SIF con condensación.

Demostración. Ver (Sabogal y Arenas, 2011, Teorema 4.2.5) □

Ejemplo 1.2.26. Considere el siguiente SIF con condensación:

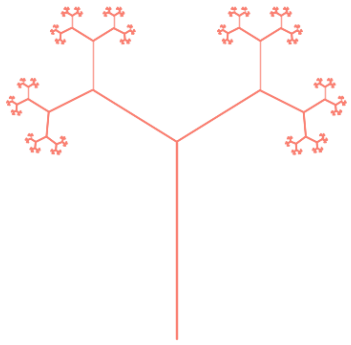
$$f_0(z) = \{0\} \times [0,1] \tag{1.22}$$

$$f_1(z) = \frac{1}{2} z e^{-i\frac{\pi}{4}} + i \tag{1.23}$$

$$f_2(z) = \frac{1}{2} z e^{i\frac{\pi}{4}} + i \tag{1.24}$$

Figura 1.9

Atractor de un SIF con condensación



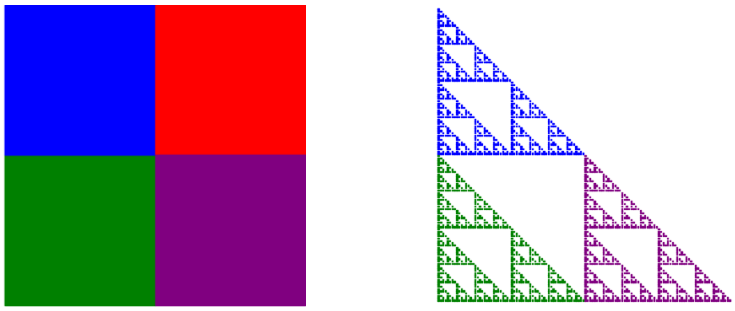
Note que la diferencia entre un SIF convencional y el SIF con condensación, es el uso de una contracción constante y como se mostró en el teorema anterior la convergencia de un SIF con condensación se comporta igual a la de un SIF.

2. El laboratorio y el museo: aspectos experimentales

En está sección nos proponemos y planteamos la experimentación sobre el comportamiento del atractor de un SIF al solo tomar elementos de un lenguaje determinado por un autómata finito determinista (AFD) . Esta idea surge del espacio de códigos asociado a un SIF ya que teniendo un alfabeto Σ y todas sus posibles combinaciones (Σ^*) resulta natural utilizar la teoría de lenguajes donde se puede construir subconjuntos a partir de procesar sus palabras en un AFD. La primera aproximación fue considerar el SIF que tiene como atractor el cuadrado relleno.

Figura 2.1

Atractores que inspiraron el experimento



Si consideramos solo las 3 primeras funciones del SIF (f_1, f_2, f_3) su atractor es el triángulo de Sierpinski, en lenguaje formales como su espacio de códigos asociado está definido sobre el alfabeto $\Sigma = \{1,2,3,4\}$, quitar f_4 se puede ver como aplicar el lenguaje $L = \{u \in \{1,2,3\}^*\}$. Para crear un código de programación que permita visualizar estos atractores debemos inspeccionar las bases teóricas con el fin extrapolar estos conocimientos a una rutina en SageMath que para este trabajo se tomó la documentación de su pagina oficial y nos referiremos a él como Sage, el primer paso fue considerar las siguientes indagaciones.

- ¿Como definir una correspondencia entre las funciones y el espacio de códigos?.
- Si tenemos una palabra se debe desarrollar un algoritmo que permita recrear la transición sobre el autómata ¿como definimos estados de aceptación y transición? y adicionalmente que funcione para cualquier autómata.
- Generar todas las palabras posibles para evaluar en los procedimientos, es decir como generar a Σ^* en código.

Para desarrollar gran parte de lo anterior consideramos hacer el código de programación utilizando arreglos de vectores, ya que nos proporciona la asignación de los elementos según la posición

que ocupen en el vector. Por ejemplo:

$$[f_1, f_2, f_3, f_4]$$

En el vector f_1 ocupa la posición 0 (En Sage las posiciones inician desde 0), f_2 ocupa la posición 1, f_3 ocupa la posición 2 y f_4 ocupa la posición 3. A continuación mostraremos el código y parte de su explicación.

2.1. Código en SageMath

```
def pemu(base,largo)
AA=range(base)
final=[]
mundo=cartesian_product([AA]*largo)
for awa in mundo:
    final.append(awa)
return(final)
```

Esta parte va generar todas las posibles combinaciones entre letras de un alfabeto. De entrada se deben seleccionar la `base` y el `largo`, donde `base` define con cuantos símbolos se usarán y `largo` será el número de veces que hará el producto cruz consigo mismo. Si $A = \{0,1,2,3\}$ y quiero tener $A \times A \times A$ entonces en código es `pemu(3,3)`.

```
def delta(palabra,Maq):
lu=0
for mi in palabra:
    lu=Maq[lu][mi]
return Maq[lu][4]
```

Consideramos escribir los autómatas como un vector de n datos, donde n es la cantidad de estados y en cada estado si usamos un alfabeto de k símbolos entonces tendrán $k+1$ componentes, donde a sus primeras k posiciones tendrá números entre 0 y k, según la posición que ocupe indicará hacía que estado deben seguir su transición y su ultimo componente indica si será aceptado o no, tomando como 1 o 0 respectivamente.

$$[\overbrace{n_1}^{a_1}, \underbrace{\overbrace{n_2}^{a_2}}_{\text{Envia al estado } n_2} \ldots, \overbrace{n_k}^{a_k}, \overbrace{(0 \vee 1)}^{\text{aceptación}}]$$

Por ejemplo considere el autómata de la figura 1,1 su representación en código es la siguiente:

$$[\overbrace{[0,0,0,1,0]}^{\text{Estado } q_0}, \overbrace{[1,1,1,2,1]}^{\text{Estado } q_1}, \overbrace{[2,2,2,2,0]}^{\text{Estado } q_2}]$$

Si tomamos en el estado q_0, la posición 3 (en Sage) diremos que si encuentra una d lo envía al estado q_1. Siempre se tomará el estado q_0 como estado inicial, las entradas del código serian: la cadena a evaluar y en el autómata que se define como el ejemplo anterior.

```
def pintapal(pal):
rutet=matrix([[1/2,0,0,1/2,0,0],[1/2,0,0,1/2,0,1],[1/2,0,0,1/2,1,0],
[1/2,0,0,1/2,1,1]])
mina=[vector([0,0]), vector([0,1]), vector([1,1]),vector([1,0])]
for uu in pal:
    tu=matrix([[rutet[uu][0],rutet[uu][1]],
    [rutet[uu][2],[rutet[uu][3]]])
    ya=vector([rutet[uu][4], rutet[uu][5]])
    lina=[]
    for ju in mina:
        sola=tu*ju+ya
        lina.append(sola)
    mina=lina
return lina
```

Definimos como se aplica el SIF, primero representamos el SIF como un arreglo de vectores donde el número de posición de cada vector representa una función del SIF, por ejemplo; en el código de la izquierda a `rutet` se le asigna el SIF de 1.5. Donde a cada f_i del SIF se organiza sus componentes como sigue:

$$f_i(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = [a,b,c,d,e,f]$$

`mina` viene siendo la "semilla"del SIF que más adelante se dibuja con la función `polygon` de Sage (Es un cuadrado de lado 1 para esta caso). El algoritmo que se plantea es que dada una palabra y por cada uno de sus caracteres se aplicará una función del SIF en la semilla, que lo determina la posición que defina el carácter de la palabra.

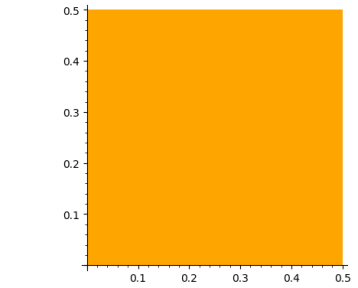
Ejemplo 2.1.1. Si se escribe `pintapal([0,2,1])` su aplicación inicial es la función que ocupe la posición 0 en `rutet`, en este caso es `[1/2,0,0,1/2,0,0]` que es la transformación que reduce en $\frac{1}{2}$ a la semilla, luego la semilla pasa a ser los puntos después de aplicarse dicha transformación es decir: `[(0, 0), (0, 1/2), (1/2, 1/2), (1/2, 0)]` y así hasta acabar los símbolos de la palabra, los valores que retornan para la palabra `[0,2,1]` son:

$$[(0,0),(0,1/2),(1/2,1/2),(1/2,0)]$$
$$[(1,0),(1,1/4),(5/4,1/4),(5/4,0)]$$
$$[(1/2,1),(1/2,9/8),(5/8,9/8),(5/8,1)]$$

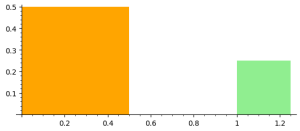
y con la función `polygon(pintapal([0,2,1])` se visualiza las transformaciones.

Figura 2.2

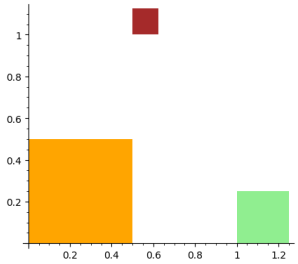
Transformación en el código



`polygon(pintapal([0])`



`polygon(pintapal([0,2])`



`polygon(pintapal([0,2,1])`

Para completar la rutina hay que añadir un bucle que una todo lo anterior y grafique cada palabra posible. El código del experimento es el siguiente:

```
def SIF(numdfun,cuantas,automata):
    colores=['green','blue','red','black']
    elpepe=[]
    gr=Graphics()
    if len(automata)==0:
        for wu in pemu(numdfun,cuantas):
            gr+=polygon(pintapal(wu),thickness = 1,color=colores[wu[len(wu)-1]])
        show(gr,axes=false)
    else:
        for unu in pemu(numdfun,cuantas):
            if delta(unu,automata)==1:
                elpepe.append(unu)
        for ewe in elpepe:
            gr+=polygon(pintapal(ewe),thickness = 1,color=colores[ewe[len(ewe)-1]])
        show(gr,axes=false)
SIF(4,8,[]) #Imprime atractor original
SIF(4,8,[[1,5,2,5,0],[1,3,2,5,1],[1,5,2,4,1],[1,5,2,5,1],[1,5,2,5,1],[5,5,5,5,0]])
```

La rutina que junta las partes anteriormente expuestas es `SIF(numdfun,cuantas,automata)`, cuyas dos primeras entradas se utilizan para iniciar `pemu` y `automata` será el autómata que se usará para determinar el lenguaje y en caso de no querer aplicar un lenguaje se debe dejar un vector vacío `[]`, su uso se muestra en las lineas 16 y 17 del código, el lector con experiencia en programación notará que es un código que no requiere de temas avanzados para recrearlo e incluso se puede mejorar.

2.2. Experimentación

Definido el código que nos permitirá graficar el comportamiento de un SIF al aplicar un autómata, presentaremos una variedad de lenguajes (regulares) aplicado a SIF's anteriormente mencionados al final del capitulo 2 y se comentará algunas observaciones. En un alfabeto $\Sigma = \{a,b,c,d\}$ definimos las siguientes asignaciones:

$f_1(x,y) = a$ con el color verde. $\quad$ $f_2(x,y) = b$ con el color azul.

$f_3(x,y) = c$ con el color rojo. $\quad$ $f_4(x,y) = d$ con el color negro.

Figura 2.3

Automata de las palabras sin bb

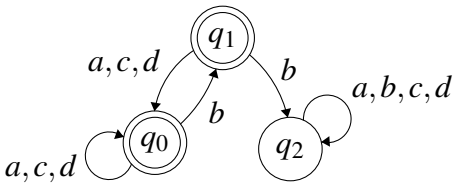


Tabla 2.1

Observaciones de las palabras sin bb

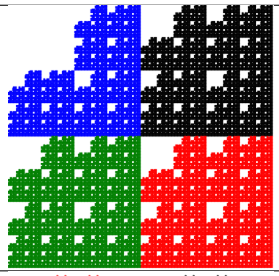
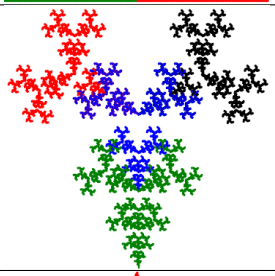
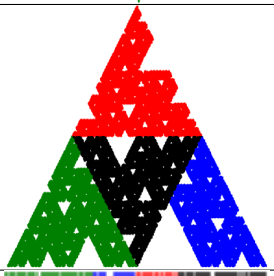
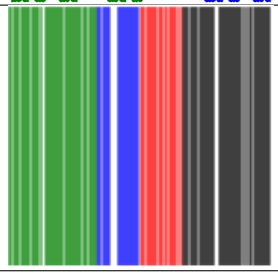
	Observaciones	
1.		Autosimilar, no habrán 2 contracciones seguidas que envíen a la esquina superior izquierda en cada región del atractor.
2.		La contracción de f_2 mueve un medio en la componente y, naturalmente si no se pueden 2 b's seguidas tampoco se podrán 3 o más por lo que las transformaciones que tengan más de 2 b's, son las que se eliminan de la figura dejando el espacio en blanco en el centro.
3.		Análogo a 2, las palabras con *bb* son aquellas secuencia que tienen únicamente 2 giros de 60 grados a la derecha, es decir aplicar *bb* hace que la parte que se elimine se ubique en la región izquierdo del triangulo azul (giro de 120) y se cumple para cada región.
4.		Cada región debería de tener una copia de la región azul, pero al aumentar las iteraciones se espera que las regiones difusas sean llenadas.

Figura 2.4

Autómata de las palabras en que las d siempre siguen de c

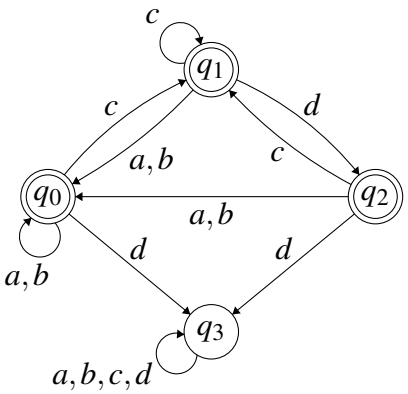


Tabla 2.2

Observaciones de las palabras en que las d siempre siguen de c

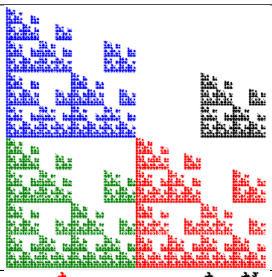
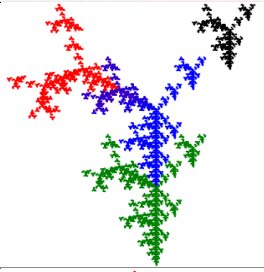
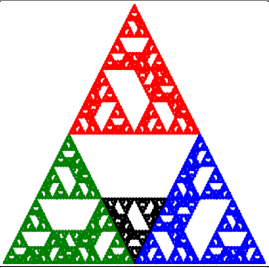
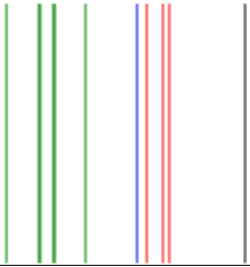
		Observaciones
5.		Presenta autosimilitud, presenta copias de la región negra en los otros. Se podría interpretar con un SIF con condensación pero ¿que región es la función constante?.
6.		Las palabras con *cd* se interpreta aquí como un giro a la izquierda seguido de uno a la derecha cada uno subiendo sobre el eje Y, por esta razón al ser consecutivos la ramificación de la derecha no presenta giro y no es tampoco el reflejo de la roja pero la región de color negro es una copia del todo.
7.		Las rotaciones dificulta la percepción de autosimilitud a primera vista, en la región negra borra las palabras con da, dd, db el cual es la región vacía en el centro de la figura.
8.		El espacio vacío entre la región roja y negra corresponde a las palabras que inician con d, la linea negra se puede entender como la palabras periódica *cd* que eventualmente se ubica en el extremo derecho ya que todas las funciones van en esa dirección.

Figura 2.5

Autómata que acepta palabras con exactamente 3 a's

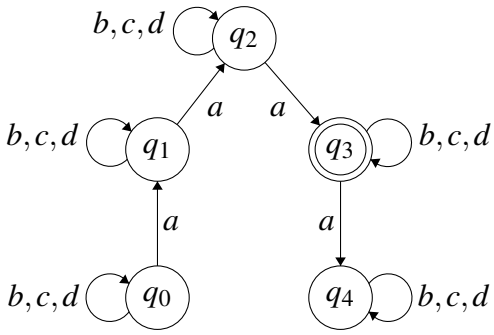


Tabla 2.3

Observaciones de las palabras que tienen exactamente 3 a's

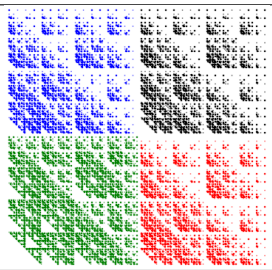
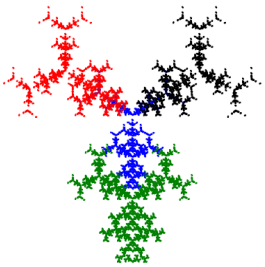
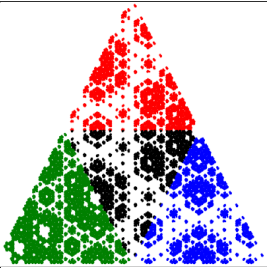
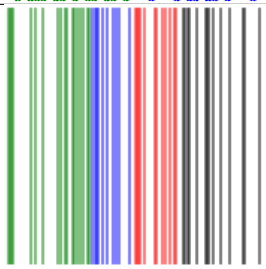
	Observaciones	
9.		Los 4 colores parecen no tener puntos en común, en la esquina inferior izquierda presenta una copia del triangulo de Sierpinski, ocurre porque en esa región es como considerar el lenguaje sin a (que ya sabemos por el ejemplo inicial del experimento) ya que no puede tomar más de 3.
10.		Sigue manteniendo la forma de su atractor original debido a que f_2 junto a f_1 se interpreta como una recta entonces limitar a 3 a's pierde la punta ubicada en la parte de abajo en la región verde, además la región roja y negra al recubrir la región azul no pierden la punta que se ve abajo en la región verde, entonces no es completamente autosimilar.
11.		Las regiones azul, negra y roja son una copia de la otra, la región verde aparenta tener la misma estructura de las demás regiones, al solo usar 3 veces f_1 los puntos que la conforman ocupan más espacio
12.		Dificíl de ver propiedades

Figura 2.6

Autómata de las palabras con un número impar de d's

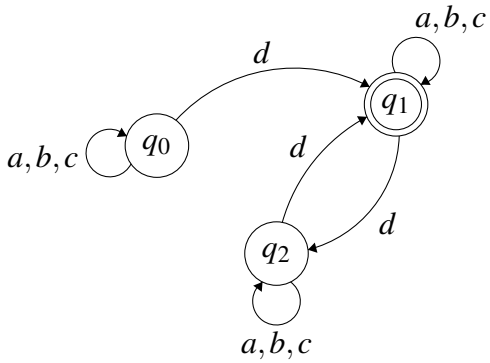


Tabla 2.4

Observaciones de las palabras con un número impar de d's

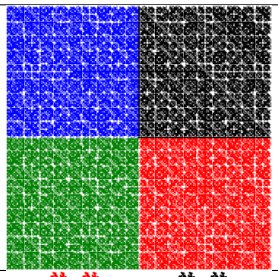
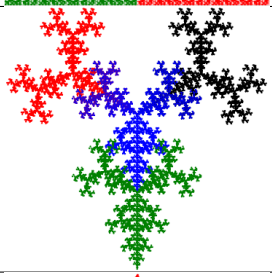
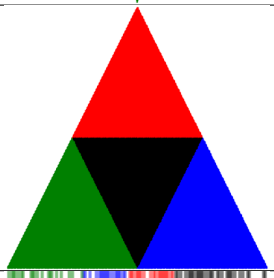
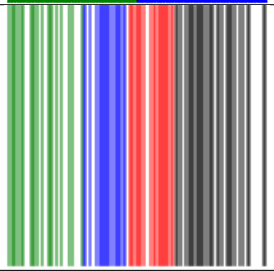
	Observaciones	
13.		Intenta llenar todo el atractor original, parece que las diagonales son más oscuras.
14.		Sin cambios.
15.		Se cree que al tomar un número par o impar de *d*'s eventualmente llegarán al atractor original de sus respectivos SIF's, un aspecto teórico a considerar.
16.		Siguiendo la tendencia de las anteriores figuras, si se aumentan las iteraciones irá llenando su espacio.

Figura 2.7

Autómata de las palabras con una d

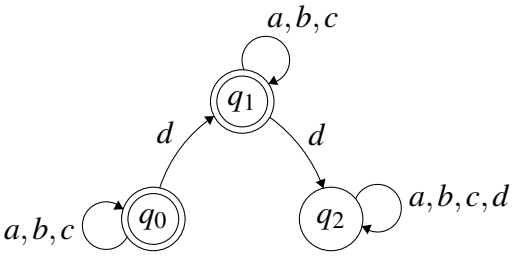
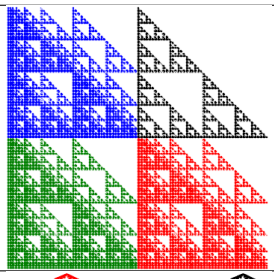
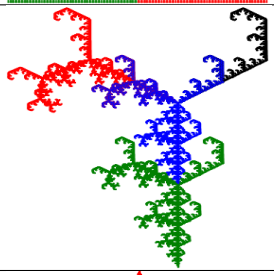
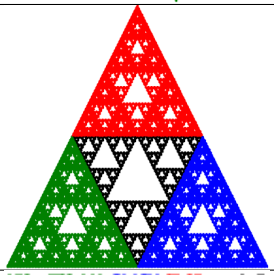
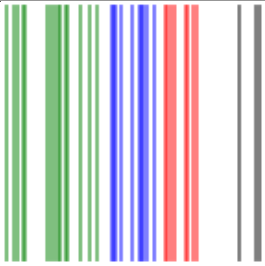


Tabla 2.5

Observaciones de las palabras con una d

	Observaciones	
17.		Presenta copias del triangulo de Sierpinski, pero con lo observado no se puede asegurar que todo sean copias de este atractor un aspecto teórico que se pueda abordar es considerarlo como un SIF con condensación cuya transformación constante es el triangulo de Sierpinski.
18.		Pierde la ramificación de la derecha, esto es debido a que solo permite un giro en esa dirección y lo usa para fabricar esta rama, conserva autosimilitud
19.		Pierde la diferenciación de hacía donde gira y presenta el triangulo de Sierpinski en el centro.
20.		No se evidencia un comportamiento similar a los atractores de este lenguaje.

El autómata es tomado de (Prusinkiewicz y Hammel, 1991)

Figura 2.8

Autómata de las palabras de la forma $(a \vee b)^*(c \vee d)^*$

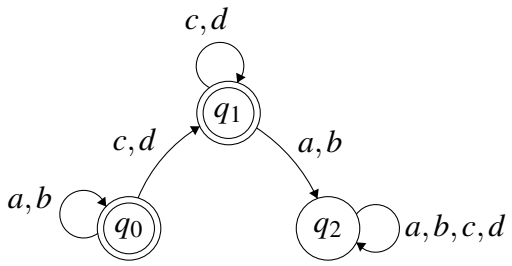


Tabla 2.6

Observaciones de las palabras de la forma $(a \vee b)^*(c \vee d)^*$

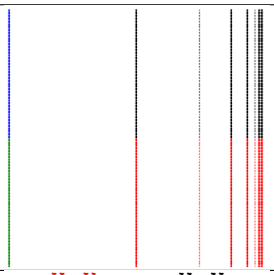
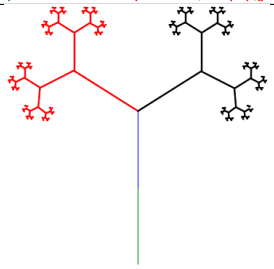
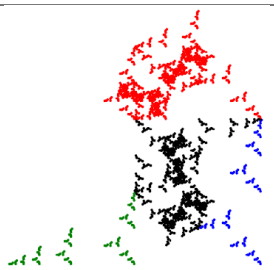
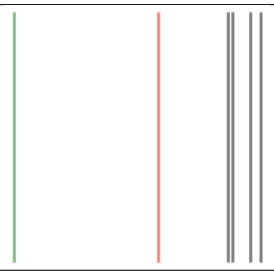
	Observaciones	
21.		El atractor de $(a \vee b)^*$ es una linea vertical y luego aplicar $(c \vee d)^*$ son movimiento hacia la derecha.
22.		Similar al caso anterior, el cambio está en f_3 y f_4 que son giros hacia la izquierda o derecha, pero este atractor si podemos afirmar que es un SIF con condensación como se ve en el ejemplo 1.9
23.		No es autosimilar y no se evidencia suficientes detalles.
24.		Pierde la región azul, se cree que es debido a que las otras funciones tienen un movimiento mas pronunciado.

Figura 2.9

Autómata de las palabras en que las b's y c's no siguen de b's o c's

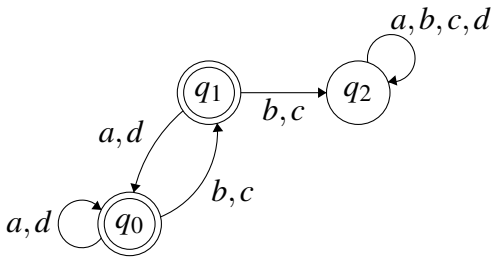
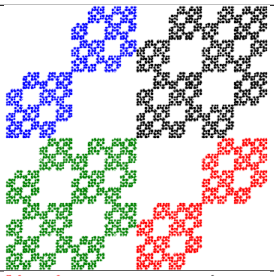
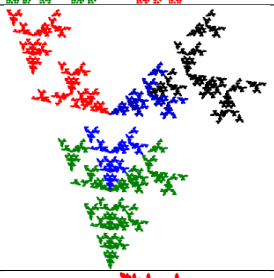
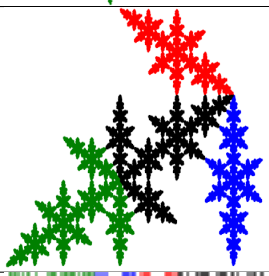
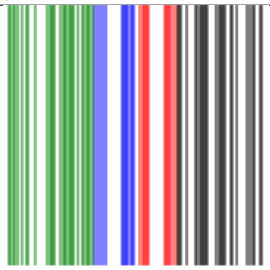


Tabla 2.7

Observaciones de las palabras en que las b's y c's no siguen de b's o c's

	Observaciones	
25.		Una interpretación sobre este SIF con lenguaje, es que tiene total movimiento en dirección de la diagonal verde-negra pero con algunas intercalaciones de movimiento sobre la diagonal azul-roja, esta conectado por 3 puntos los cuales son en la región azul, la intersección verde-negra y en la región roja.
26.		Pierde las combinaciones que tengan algún giro de 45 a la izquierda, pero esta se puede intercalar y por eso se alcanza a dividir hacía la región
27.		Tiene una simetría en la región verde-negro porque estas palabras son afectadas por un giro de 180, la región roja y la azul son copias de la región verde negro, los picos que aparecen son productos de intercalar giros de 60 grados a la izquierda ó hacia la derecha (pero no seguido)
28.		Como intercala entre *b*'s y *c*'s entonces las transformaciones entre estos dos símbolos da la región vacía del centro.

El autómata es tomado de (Prusinkiewicz y Hammel, 1991)

Figura 2.10

Autómata de las palabras sin una letra consecutiva

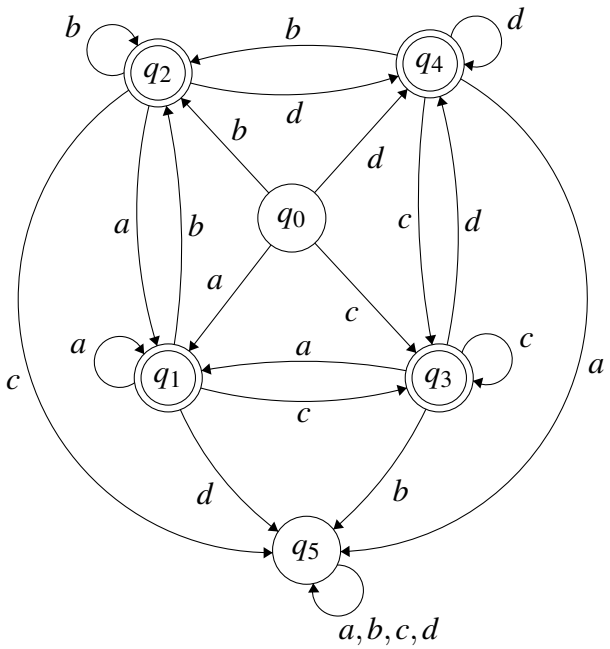


Tabla 2.8

Observaciones de las palabras sin una letra consecutiva

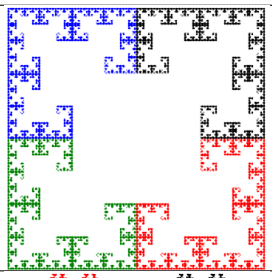
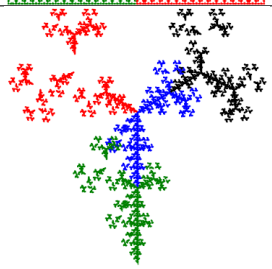
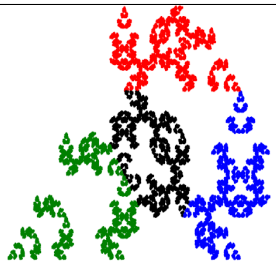
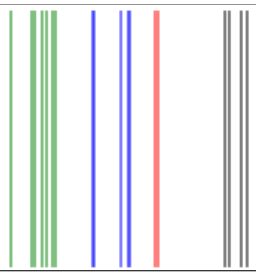
		Observaciones
29.		Se evidencia autosimilitud en cada esquina pero no en el todo.
30.		No parecen evidenciarse copias entre las regiones ni con el todo.
31.		Tiene simetría diagonal, la región verde presenta copias de la roja-azul e incluso se podría decir que la azul y roja tienen copias del todo dentro de si mismas, pero la región negra parece un caso particular.
32.		Sin comentarios.

Figura 2.11

Autómata de las palabras en que las b's que siguen de a′s o d's que siguen de c's

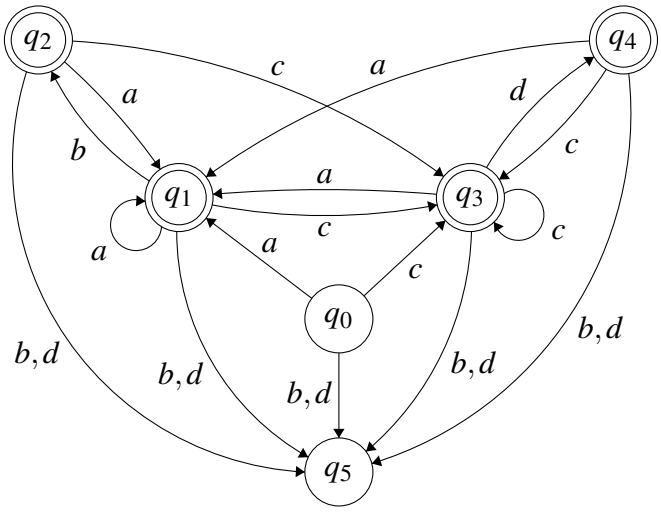


Tabla 2.9

Observaciones de las palabras en que las b's que siguen de a's o d's que siguen de c's

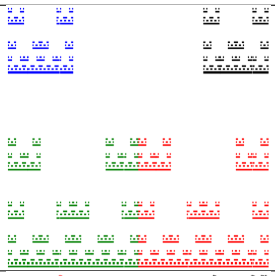
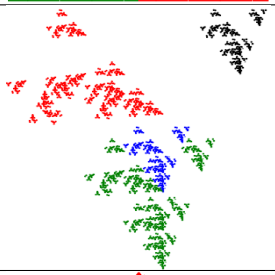
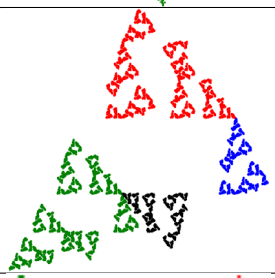
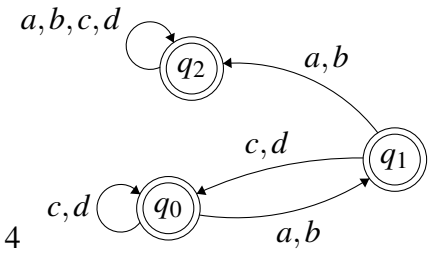
	Observaciones
33.	 Se evidencia autosimilitud total, cada región es una copia del todo.
34.	 Cada región es una copia del todo, pero la región negra y azul son más pequeñas por la condición de tener obligatoriamente un prefijo especifico
35.	 Análogo a los casos anteriores, son copias de cada uno y la región negra y azul son mas pequeñas.
36.	 Elimina totalmente la región azul y negra, esto es por la manera en que se asgina los codigos, es decir solo quedaron las palabras que inician con a o c.

Figura 2.12

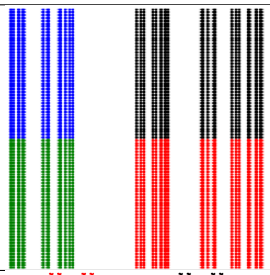
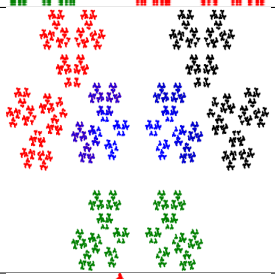
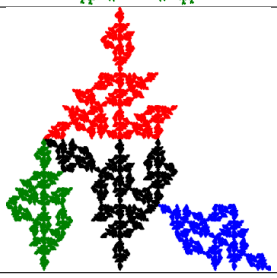
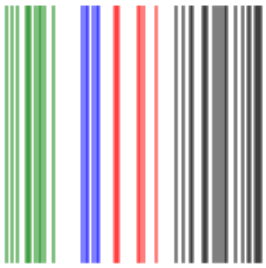
Autómata de las palabras en que las a's y b's no siguen de a's o b's



4

Tabla 2.10

Observaciones de las palabras en que las a's y b's no siguen de a's o b's

	Observaciones	
37.		La región de las palabras con *a*'s y *b*'s seguidas al no ser aceptada quedará vacía, porque a pesar de que en un primer vistazo este atractor no parece coincidir con el lenguaje es porque la región de la izquierda queda eliminada.
38.		Las palabras en que las *b*'s y *a*'s siguen de *b*'s o *a*'s, era la linea del centro del atractor original, eliminarla solo nos deja con los giros y la ausencia de esta crea una simetría con respecto al vacío que forma.
39.		La región roja es una copia de la negra y es a su vez son copia del todo, la verde y la azul también son copias entre ellas pero no del todo.
40.		La región difusa en el color verde son las palabras que se intercalan con *a*'s, en la región azul se pierde totalmente las palbras que inician con *ba*, en la roja se desaparecen las palabras con *cba*, *cbb*, *cab*, *caa* por eso se ve más vacío y en la región negra se ve mens afectada debido a que la regla la incluye como ocurre en la región verde.

3. Primera formalización

Experimentalmente este proceso iterativo consigue visualizar la aplicación de los lenguajes sobre los atractores, cada una de las anteriores figuras desprenden información que naturalmente dependen de la elección del SIF y de su orden entre las funciones. Teniendo en cuenta las observaciones propondremos un análisis teórico que intentará explicar el comportamiento de los atractores mostrados, en esta sección intentaremos formalizar las observaciones.

Definición 3.0.1. Que v sea prefijo de w que se denota $v < w$, significa que existe una palabra u tal que $w = vu$.

Proposición 3.0.2. Σ^* *queda parcialmente ordenado por* $<$ *y cumple:*

1. *Si* $v < w$ *entonces* $uv < uw$*;*
2. *Si* $u < v$ *y* $w < v$ *entonces* $u < w$ *o bien* $w < u$*;*
3. *Todo subconjunto de palabras acotado, es una cadena finita y por tanto tiene un máximo.*

Demostración. 1. De la definición.

2. Sean $v = a_1a_2\ldots a_k$, si $u < v$ entonces existe un w tal que $u = a_1a_2\ldots a_n$ y $u = a_1a_2\ldots a_m$ donde $n \leq k$ y $m \leq k$ y se tiene que $n \leq m$ o $m \leq n$ según sea el caso $u < w$ o $w < u$.

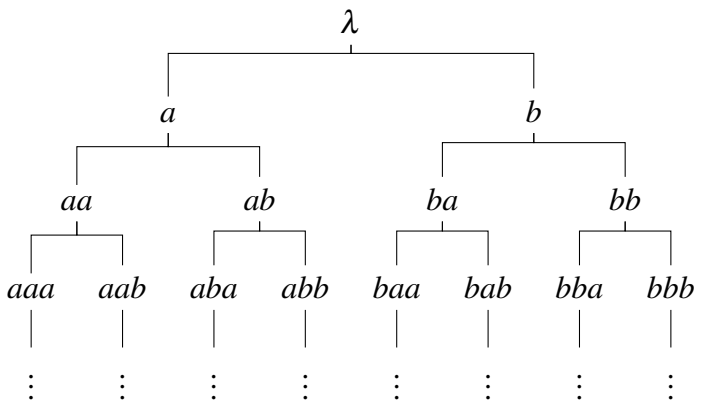
3. por 2. todo subconjunto acotado es una cadena. Cada palabra de longitud n tiene a lo más $n+1$ elementos .

□

De la proposición anterior decimos que la propiedad 1. indica que el orden cumple la concatenación por izquierda. 2. asegura que $(\Sigma^*, <)$ es un orden expansivo.

Figura 3.1

Árbol para el alfabeto binario



Con el orden anteriormente definido podemos formar arboles similares a como se ve en la figura 3.1. Estos arboles se asemejan a como se construyen los fractales en el experimento, se consideran todas las posibles longitudes de palabras formadas por el alfabeto, para la teoría clásica los fractales son el limite del proceso iterativo de recorrer todas las ramas, si queremos acercarnos a este concepto utilizando lenguajes debemos definir palabras limite dentro del lenguaje, es decir en el árbol veremos que ramas alcanzan la copa.

Definición 3.0.3. Las cadenas maximales se llamarán códigos, el conjunto de todos los códigos se denota por: $\Sigma^{\mathbb{N}}$.

Definición 3.0.4. La unión (disjunta) de las palabras y los códigos sobre un alfabeto finito Σ, constituye el espacio de los discursos que será denotado por $\Sigma^{\nabla} = \Sigma^{\mathbb{N}} \cup \Sigma^{*}$. El orden se extiende de la siguiente manera:

1. si p y q son palabras $p \leq q$ si y solo si $p < q$
2. si $p \in \Sigma^{*}$ y $q \in \Sigma^{\omega}$ entonces $p \leq q$ siempre que existe $\sigma \in \Sigma^{\omega}$ tal que $q = p\sigma$
3. Si p y q son códigos $p \leq q$ si y solo si $p = q$

Esta extensión del orden nos permite relacionar los códigos con las palabras, en que un código es mayor a una palabra pero no al revés.

Definición 3.0.5. Definimos en Σ^{∇} la topología de orden $\tau_{\leq}$, cuyos abiertos subbásicos, son los rayos

$$(u, +\infty) = \{q \in \Sigma^{\nabla} \mid u < q\} \text{ y}$$
$$(-\infty, v) = \{q \in \Sigma^{\nabla} \mid q < v\}.$$

Si $u \in \Sigma^{*}$, entonces :

$$\{u\} = \begin{cases} (-\infty, a) & \text{si } u = \lambda; \\ (a_1 a_2 \cdots a_{k-1}) \cap (-\infty, ua) & \text{si } u = a_1 a_2 a_3 \cdots a_k. \end{cases}$$

De lo anterior $\{u\} \in \tau_{\leq}$ para cada $u \in \Sigma^{*}$

Proposición 3.0.6. *El espacio Σ^{∇} dotado de la topología de orden, satisface las siguientes condiciones:*

1. *Es compacto.*
2. *Es totalmente disconexo.*

3. Σ^* *es un subconjunto denso, donde* $\Sigma^* \cong \mathbb{N}$. *Esto es,* Σ^∇ *es una compactación de* $\mathbb{N}$ *con residuo* $\Sigma^\mathbb{N}$.

4. $\Sigma^\mathbb{N}$ *es metrizable.*

5. Σ^∇ *es homeomorfo al subconjunto del plano*

Note que si se considera $\Sigma^\mathbb{N}$ como subespacio de Σ^∇ coincide con el espacio de 1.2.1.

Ejemplo 3.0.7. Si L es el lenguaje a^*b, donde $\Sigma = \{a,b\}$, entonces $L \subset \Sigma^*$ donde $\bar{L} \cap \Sigma^\mathbb{N} = \{(p_i)_{i=1}^\infty \Sigma^\mathbb{N} \mid p_i = a\}$

Definición 3.0.8. Denotaremos por $\tilde{L}$ al conjunto $\bar{L} \cap \Sigma^\mathbb{N}$.

Proposición 3.0.9. *Si L es un lenguaje finito, entonces* $L = \bar{L}$ *y por tanto* $\tilde{L} = \emptyset$

Nota: Dado un lenguaje L, entonces $\tilde{L}$ determina un subconjunto cerrado de $\Sigma^\mathbb{N}$.

Teorema 3.0.10. *Sea* $\mathscr{A}$ *el atractor de un SIF* $\{X, f_1, f_2, \ldots, f_N\}$ *y* $\varphi : \Sigma^\mathbb{N} \to \mathscr{A}$ *la función direccionamiento. Si L es un lenguaje, entonces L induce un compacto sobre* $\mathscr{A}$ *determinado por* $\varphi(\tilde{L})$ *representa un compacto en* $\mathscr{A}$.

Demostración. $\tilde{L}$ determina un conjunto cerrado en $\Sigma^\mathbb{N}$, como la función dirección φ es una función continua y cerrada entonces $\varphi(\tilde{L})$ es un conjunto cerrado, luego al ser un subconjunto cerrado de un espacio compacto por (Willard, 2004, Teorema 17.5) se concluye que $\varphi(\tilde{L})$ representa un compacto en $\mathscr{A}$. □

4. Segunda formalización

En un principio se busca contrastar la teoría conocida sobre los atractores de SIF's con los resultados observados al limitarlo con un lenguaje. La notación e ideas fueron tomadas de (Prusinkiewicz y Hammel, 1991), quienes añaden más contenido sobre el estado de transición del autómata y aplican un homomorfismo entre las funciones del SIF y el alfabeto, esta caracterización es similar a como se elabora el código en Sage que se usa para visualizar los atractores y finalizan con una condición que garantiza la clausura de estos atractores.

Definición 4.0.1. Es llamado el lenguaje asociado a un estado s_k al conjunto

$$L(s_k) = \{w \in \Sigma^* \mid \delta(s_0, w) = s_k\}$$

Definición 4.0.2. El lenguaje aceptado por un autómata M se define como

$$L(M) = \bigcup_{s_k \in E} L(s_k) = \{w \in \Sigma^* \mid (\exists s_k \in E)\delta(s_0, w) = s_k\},$$

donde E es el conjunto de estados de aceptación.

Lema 4.0.3. *Para algún estado s_k distinto al inicial, la siguiente igualdad se cumple*

$$L(s_k) = \bigcup_{\delta(s_i,a)=s_k} L(s_i)a \tag{4.1}$$

En caso de querer añadir al estado inicial se tiene como sigue:

$$L(s_0) = \bigcup_{\delta(s_i,a)=s_0} L(s_i)a \cup \varepsilon. \tag{4.2}$$

Ejemplo 4.0.4. Considere el autómata de las palabras de la forma $(a \vee b)^*(c \vee d)^*$, los lenguajes asociados a los estados y el autómata son como sigue:

$$\begin{aligned} L(s_0) &= L(s_0)(a \cup b) \cup \varepsilon \\ L(s_1) &= L(s_0)(c \cup d) \cup L(s_1)(c \cup d) \\ L(M) &= L(s_0) \cup L(s_1) \end{aligned}$$

Llaman a la aplicación de un lenguaje sobre un SIF como *language-restricted iterated funtion system (LRIFS)*. Proponemos que la traducción de lo anterior sea: Restricción de sistemas iterados de funciones o R-SIF y se define como una quíntupla $\mathscr{F}_L = \langle X, F, V, h, L\rangle$, donde:

- X un espacio métrico completo,

- F un SIF sobre el espacio métrico,
- V un alfabeto,
- $h : V \to F$ envía letras del alfabeto al SIF F,
- $L \subset V^*$ un lenguaje sobre V.

El dominio de h se puede extender al conjunto de palabras V^* usando la siguiente ecuación

$$h(a_1 a_2 \dots a_n) = h(a_1) \circ h(a_2) \circ \cdots \circ h(a_n).$$

Esta extensión de h es un homomorfismo entre el monoide generado por V^* con la concatenación y el monoide generado por el SIF con la composición. La imagen de L está dada por la siguiente ecuación:

$$h(L) = \bigcup_{w \in L} h(w). \tag{4.3}$$

Si L es el lenguajes de todas las palabras sobre el alfabeto, es decir $L = V^*$ entonces $h(V^*)$ es el conjunto de todas las transformaciones del SIF definido como sigue

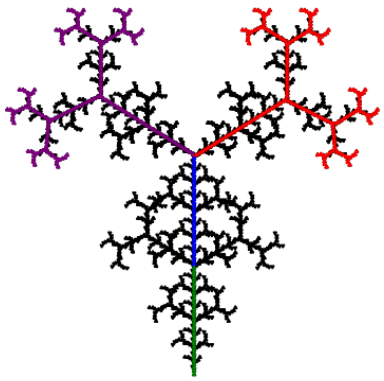
$$h(L) = (F_1 \vee F_2 \vee \cdots \vee F_n)^* = S^*.$$

A partir de aquí los autores involucran un termino $x_0 \in \mathscr{A}$ donde su imagen con respecto a la transformación $h(L)$ se denota por $\mathscr{A}_L(x_0)$, que utilizan para mostrar que es $\mathscr{A}_L(x_0)$ es un subconjunto del atractor original.

$$\mathscr{A}_L(x_0) = x_0 \circ h(L) \subset x_0 \circ h(V^*) = S^*(x_0) = \mathscr{A}$$

Figura 4.1

De fondo el atractor original y en colores el lenguaje $(a \vee b)^*(c \vee d)^*$



En esta formalización su contenido es adecuado para interpretar la ejecución del código de programación en Sage, el homomorfismo es análogo a la asignación posicional de las funciones y la imagen de un lenguaje en la ecuación 4.3 es comparable a la rutina que se usa para graficar todas las palabras del lenguaje. Los autores parten del lema 4.0.3 para describir la estructura recursiva de estos atractores afectados por un lenguaje. Considere $\mathscr{F}_L$ un R-SIF y un autómata que acepte L se tiene que

$$L(s_0) = \bigcup_{\delta(s_i,a)=s_0} L(s_i)a \cup \varepsilon \tag{4.4}$$

$$L(s_k) = \bigcup_{\delta(s_i,a)=s_k} L(s_k)a \tag{4.5}$$

Después aplicando el homomorfismo h en ambos lados para algún $x_0 \in X$ arbitrario se obtiene que:

$$x_0h(L(s_0)) = \bigcup_{\delta(s_i,a)=s_0} x_0h(L(s_i))h(a) \cup \{x_0\} \tag{4.6}$$

$$x_0h(L(s_k)) = \bigcup_{\delta(s_i,a)=s_k} x_0h(L(s_i))h(a) \tag{4.7}$$

y tomando $\mathscr{A}_k = x_0h(L(s_k))$ la imagen del conjunto $\{x_0\}$ con respecto a la transformación $h(L(s_k))$ y suponga $F_a = h(a)$ se obtiene que:

$$\mathscr{A}_0 = \bigcup_{\delta(s_i,a)=s_0} \mathscr{A}_i \circ F_a \cup \{x_0\} = \bigcup_{\delta(s_i,a)=s_0} F_a(\mathscr{A}_i) \cup \{x_0\} \tag{4.8}$$

$$\mathscr{A}_k = \bigcup_{\delta(s_i,a)=s_k} \mathscr{A}_i \circ F_a = \bigcup_{\delta(s_i,a)=s_k} F_a(\mathscr{A}_i) \tag{4.9}$$

Construyeron una ecuación similar al ejemplo 1.2.13, este sistema describe como actúa el lenguaje en un SIF, el R-SIF divide el atractor en un número finito de subconjuntos $\mathscr{A}_k$ y cada uno de ellos es la unión de los $\mathscr{A}_i$ con respecto a la transformación F_a, pero las ecuaciones 4.6 y 4.7 incluyen el termino x_0 que representa el punto inicial y afirman que la elección de este punto afecta $\mathscr{A}_L(x_0)$, sin embargo en la teoría conocida sobre SIF's su atractor será el mismo sin importar el elemento que se tome, entonces se decidió explorar en el experimento tomar como semilla puntos en distintas ubicaciones y no se manifestaron cambios en los atractores, no es claro el uso de esta dependencia del punto inicial. Finalizan esta sección con la comparación de la ecuación:

$$\mathscr{A} = \bigcup_{F_i \in \mathscr{F}} F_i(\mathscr{A}) \tag{4.10}$$

En que el atractor de un SIF se puede entender como el conjunto cerrado mas pequeño que la

cumple, pero en los R-SIF afirman que hay lenguajes que no siempre poseen el conjunto cerrado más pequeño y consideran de ejemplo el lenguaje $(a^* \vee b^*)$ sobre una recta donde $\mathscr{A}_L$ son dos puntos aislados y finalizan esta sección dando una condición suficiente para la existencia del conjunto compacto mas pequeño generado por un R-SIF.

Teorema 4.0.5. *Considere un RSIF y asuma que L tiene la propiedad del prefijo (si existe una palabra no vacía $v \in \Sigma^*$ tal que $vL \subset L$) denotado por x_0 el punto fijo de la transformación $h(v)$ y dado $A = x_0 h(L)$. Entonces para cualquier $x \in X$, la inclusión $A \subset xF(L)$ se cumple.*

Demostración. De la inclusión $vL \subset L$ sigue que en $v^n L \subset L$ para algún $n = 0, 1, 2, \ldots$. por lo tanto para algún x

$$x_0 F(L) = \lim_{x \to \infty} xF(v^n)F(L) = \lim_{x \to \infty} xF(v^n L) \subset xF(L)$$

□

A pesar de que presentan un desarrollo teórico que se asemeja a la construcción del experimento, la finalización de este contenido no se esclarece lo suficiente, puntualmente en la elección del punto inicial y lo que consideran como cerrado.

5. Discusión

En este proyecto experimental se consultó las bases teóricas que permitieran plantear un código de programación que uniera lo referente a lenguajes regulares y la geometría fractal, presentamos formalizaciones que intentan establecer teóricamente las visualizaciones del experimento y se muestra algunos resultados de los antecedentes hallados sobre el tema, creemos que quedaron inquietudes y aspectos que se pueden ampliar que algún lector interesado pueda trabajar, entre esas están:

- Experimentar con la dimensión de los atractores de lenguajes.
- Caracterizar la autosimilitud de estas figuras.
- Proponer otra formalización o incluso ampliar la presentada en el proyecto.
- El problema inverso: hallar las funciones que correspondan al atractor de un lenguaje.
- Desarrollar un software que facilite el proceso de experimentar.
- Encontrar aplicaciones de lo desarrollado.
- Indagar sobre los casos en que sea fácil encontrar un SIF con igual atractor o SIF con condensación.

Referencias

Barnsley, M. F. (2014). *Fractals everywhere*. Academic press.

Camargo, J. E., y Villamizar, E. J. (2019). *Topología general*. Universidad Industrial Santander.

de Castro Korgi, R. (2004). *Teoría de la computación: lenguajes, autómatas, gramáticas*. Universidad Nacional de Colombia.

Edgar, G. (2008). *Measure, topology, and fractal geometry* (Vol. 2). Springer.

Hutchinson, J. E. (1981). Fractals and self similarity. *Indiana University Mathematics Journal*, *30*(5), 713–747.

Linz, P. (2006). *An introduction to formal languages and automata*. Jones & Bartlett Learning.

Mandelbrot, B. B. (1982). *The fractal geometry of nature*. WH freeman New York.

Prusinkiewicz, P., y Hammel, M. (1991). Automata, languages and iterated function systems. *Fractal Models in*, 115–143.

Sabogal, S., y Arenas, G. (2011). *Una introducción a la geometría fractal*. Universidad Industrial Santander.

Willard, S. (2004). *General topology*. Dover Publications.

Édgar, E. (1994). *El espacio de los códigos. monografía*. Universidad Industrial de Santander, Bucaramanga.