

Generación de curvas fractales a partir de homomorfismos entre lenguajes [con Mathematica[®]]

JOSÉ L. RAMÍREZ^{a*}, GUSTAVO N. RUBIANO^b

^a Universidad Sergio Arboleda, Escuela de Matemáticas, Bogotá, Colombia.

^b Universidad Nacional de Colombia, Depto. de Matemáticas, Bogotá, Colombia.

Resumen. En este artículo se hace una implementación con el *software Mathematica 8.0* de algunas propiedades combinatorias de la cadena o palabra de Fibonacci, la cual se puede generar a partir de la iteración de un homomorfismo entre lenguajes. Asimismo se recopilan algunas propiedades gráficas de la curva fractal asociada a esta cadena de símbolos, la cual se puede generar a partir de unas reglas de dibujo análogas a las utilizadas en los L-Sistemas. Todos los códigos utilizados en el artículo se presentan en detalle y luego se aplican para generar nuevas curvas fractales. Finalizamos con una forma alternativa de generar la curva de Fibonacci y otras curvas a partir de cadenas características.

Palabras claves: Combinatoria sobre cadenas, cadena infinita de Fibonacci, homomorfismos entre lenguajes, curvas fractales, L-sistemas, Mathematica[®].

MSC2010: 11B39, 28A80, 68R15, 97N80

Generating fractals curves from homomorphisms between languages [with Mathematica[®]]

Abstract. In this paper we implement with the software *Mathematica 8.0* some combinatorial properties of Fibonacci Word, which can be generated from the iteration of a homomorphism between languages. We collect also some graphic properties of the fractal curve associated to this word, which can be generated from drawing rules similar to those used in the L-Systems. All codes used in this paper are presented in detail and then they are applied to generate new fractal curves. We conclude with an alternative way to generate the Fibonacci curve and other curves from characteristics words.

Keywords: Combinatorics on words, infinite Fibonacci word, homomorphism between languages, fractal curves, L-systems, Mathematica[®].

* Autor para correspondencia: *E-mail:* jose.l.ramirez@ima.usergioarboleda.edu.co
Recibido: 01 de agosto de 2012, Aceptado: 14 de septiembre de 2012.

1. Introducción

En este artículo se hace una implementación con el software *Mathematica 8.0* de algunas propiedades combinatorias de la cadena infinita de Fibonacci, la cual lleva dicho nombre ya que su construcción es análoga a la definición recursiva de los números de Fibonacci. Estas propiedades son ejemplos sencillos del tipo de cosas que se estudian en la Combinatoria sobre Cadenas, la cual es una rama reciente de las matemáticas discretas, que estudia las cadenas finitas e infinitas de símbolos y tiene aplicaciones en la teoría de autómatas y lenguajes formales y en la teoría de números, entre otras. Asimismo se recopilan algunas propiedades gráficas de la curva fractal asociada a esta cadena de símbolos, la cual se puede generar a partir de unas reglas de dibujo análogas a las utilizadas en los L-Sistemas. Todos los códigos utilizados en el artículo se presentan en detalle y luego se aplican para generar nuevas curvas fractales como la de Thue-Morse y la de Baum-Sweet.

En la Sección 2 presentamos la notación y definiciones utilizadas en el desarrollo del artículo. En la sección 3 definimos la cadena de Fibonacci y establecemos su relación con los números de Fibonacci; además, definimos un homomorfismo entre lenguajes que permite en cada iteración encontrar la cadena n -ésima de Fibonacci. En la sección 4 mostramos en detalle algunas propiedades combinatorias de la cadena de Fibonacci, que luego serán utilizadas en la sección 5 para obtener información de la curva fractal de Fibonacci, la cual se obtiene usando el lenguaje LOGO o lenguaje de la Tortuga. En la sección 6 desarrollamos un código utilizando el *software Mathematica 8.0* para la implementación gráfica de esta curva y de algunas de sus propiedades. En la sección 7 mostramos que la curva de Koch se puede obtener a partir de la cadena de Thue-Morse. Finalizaremos el artículo con una forma alternativa de obtener la cadena de Fibonacci a partir de las cadenas características y mostramos su implementación en *Mathematica 8.0* que permite ampliar las posibilidades de trabajo.

2. Notación y definiciones preliminares

Las *cadenas* o *palabras* son una sucesión finita de símbolos (a_1, a_2, \dots, a_n) tomadas de un conjunto finito no vacío Σ llamado *alfabeto*. Para facilitar la escritura, la cadena se escribe simplemente como $a_1 a_2 \dots a_n$ y se denotan por las letras u, v, w, x, \dots . Se supone la existencia de una única cadena λ que no tiene símbolos llamada *cadena vacía*. Definimos Σ^* como el conjunto de todas las cadenas sobre un alfabeto Σ , incluyendo la cadena vacía. La *longitud* $|u|$ de una cadena $u \in \Sigma^*$ se define como el número de símbolos de u , incluyendo símbolos repetidos. Por $|u|_a$ representamos la cantidad de veces que aparece el símbolo a en la cadena u .

Dadas dos cadenas $u = a_1 a_2 \dots a_k$ y $v = b_1 b_2 \dots b_s$ en Σ^* , la *concatenación* uv es la cadena que resulta al escribir los símbolos de u y a continuación los símbolos de v , es decir $uv = a_1 a_2 \dots a_k b_1 b_2 \dots b_s$. Como caso particular, si $v = \lambda$ entonces $u\lambda = \lambda u = u$. Algunas propiedades básicas son $|uv| = |u| + |v|$ y $|u^n| = n|u|$ para todo entero $n \geq 0$, donde $u^n = u \dots u$ (n -veces) es llamada la n -potencia de u .

Una cadena v es una *subcadena* o *factor* de u si existen cadenas $x, y \in \Sigma^*$ tales que $u = xvy$. Es claro que x ó y pueden ser λ , así la cadena vacía es subcadena de cualquier

cadena y toda cadena es subcadena de sí misma. Si $u = vw$, entonces las subcadenas v y w se llaman *prefijo* y *sufijo* de u , respectivamente.

La *inversa* u^R de una cadena $u \in \Sigma^*$ se define como

$$u^R = \begin{cases} \lambda & \text{si } u = \lambda, \\ a_n a_{n-1} \cdots a_1 & \text{si } u = a_1 a_2 \cdots a_n. \end{cases}$$

Una cadena u es un *palíndromo* si $u^R = u$.

Una *cadena infinita* $\mathbf{u} = a_1 a_2 a_3 \dots$ es una sucesión infinita de símbolos, y se denota con letra minúscula en negrilla. El conjunto de todas las cadenas infinitas sobre Σ se denota como Σ^ω ; además, $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$.

Ejemplo 2.1. $\mathbf{p} = (p_n)_{n \geq 1} = 0110101000101 \dots$ es una cadena infinita, donde $p_n = 1$ si n es un número primo y $p_n = 0$ en caso contrario.

Definición 2.2. Sean Σ y Δ dos alfabetos. Un *homomorfismo* es una función $h : \Sigma^* \rightarrow \Delta^*$ tal que $h(xy) = h(x)h(y)$ para todo $x, y \in \Sigma^*$. Es claro que $h(\lambda) = \lambda$, y que h queda completamente determinado si se conoce $h(a)$ para todo $a \in \Sigma$.

3. La cadena de Fibonacci

Definición 3.1. La *cadena n -ésima de Fibonacci* f_n se define recursivamente por $f_0 = 1$, $f_1 = 0$ y $f_n = f_{n-1}f_{n-2}$ para todo $n \geq 2$. Definimos la *cadena infinita de Fibonacci* \mathbf{f} como

$$\mathbf{f} := \lim_{n \rightarrow \infty} f_n = 0100101001001 \dots$$

Definición 3.2. Definimos el *homomorfismo σ de Fibonacci* como $\sigma : \{0, 1\}^* \rightarrow \{0, 1\}^*$ donde $\sigma(0) = 01$ y $\sigma(1) = 0$.

El siguiente teorema permite generar la cadena de Fibonacci a partir de la iteración del homomorfismo σ . Esta relación es clave para su implementación gráfica en *Mathematica 8.0*.

Teorema 3.3. La *cadena \mathbf{f} de Fibonacci* satisface que

$$\lim_{n \rightarrow \infty} \sigma^n(1) = \mathbf{f}.$$

Demostración. Demostraremos que $\sigma^n(1) = f_n$ y $\sigma^n(0) = f_{n+1}$ para todo $n \geq 0$. El argumento es por inducción sobre n . Para $n = 0$, $y_n = 1$ se tiene claramente. Supongamos que se tiene para n :

$$\begin{aligned} \sigma^{n+1}(1) &= \sigma^n(\sigma(1)) = \sigma^n(0) = f_{n+1}, \\ \sigma^{n+1}(0) &= \sigma^n(\sigma(0)) = \sigma^n(01) = \sigma^n(0)\sigma^n(1) = f_{n+1}f_n = f_{n+2}. \end{aligned}$$

De lo anterior se concluye que $\lim_{n \rightarrow \infty} \sigma^n(1) = \mathbf{f}$. □

En general, el estudio de las propiedades de las cadenas finitas e infinitas de símbolos hace parte de una rama de las matemáticas llamada combinatoria sobre cadenas. Esta se inicia en el año 1906, con los trabajos del matemático Axel Thue sobre repetición de cadenas (ver, e.g., [11]). Su estudio se iniciaría de manera sistemática hacia el año 1950 con el matemático francés Marcel-Paul Schützenberger y sus estudios sobre teoría de códigos [20]. Sin embargo, el estudio de la combinatoria sobre cadenas se consolidaría con la publicación en el año 1983 del libro *Combinatorics on Words* el cual encierra el trabajo de varios investigadores bajo el seudónimo de Lothaire. Con este mismo seudónimo aparecieron dos volúmenes más en los años 2002 y 2005, consolidándose así esta línea de investigación, (ver [12] y [13]).

Proposición 4.1. *Las cadenas 11 y 000 no son subcadenas de la cadena de Fibonacci.*

Demostración. El argumento es por inducción sobre n . Si $n = 0$ entonces $f_0 = 1$. Supongamos que se tiene para todo entero menor o igual que n ; entonces, como $f_{n+1} = f_n f_{n-1}$, por la hipótesis de inducción f_n y f_{n-1} no tienen como subcadena a 11, luego la única posibilidad que queda es que 1 sea un sufijo de f_n y un prefijo de f_{n-1} ; pero esto es una contradicción, ya que toda cadena f_n con $n \geq 1$ comienza con 0. El argumento es análogo para la subcadena 000. \square

Proposición 4.2. *Sea ab un sufijo de la cadena de Fibonacci f_n . Entonces, para todo $n \geq 2$, $ab = 01$ si n es par y $ab = 10$ si n es impar (i.e., sólo existen dos sufijos de longitud 2).*

Demostración. El argumento es por inducción sobre n . Si $n = 2$, entonces $f_2 = 01$; si $n = 3$, entonces $f_3 = 010$. Asumamos que se tiene para todo entero menor o igual que n . Si $n + 1$ es par, entonces, como $f_{n+1} = f_n f_{n-1}$ y $n - 1$ es par, se tiene por hipótesis de inducción que f_{n-1} finaliza en 01 y por lo tanto f_{n+1} también. Análogamente si $n + 1$ es impar. \square

Proposición 4.3. *$f_{n-1}f_{n-2}$ y $f_{n-2}f_{n-1}$ tienen un prefijo común de longitud $F_n - 2$ para todo $n \geq 3$ (i.e., $f_{n-1}f_{n-2}$ y $f_{n-2}f_{n-1}$ son iguales salvo por los dos últimos símbolos).*

Demostración. Por definición de f_n se tiene que

$$\begin{aligned} f_{n-1}f_{n-2} &= f_{n-2}f_{n-3}f_{n-3}f_{n-4} = f_{n-3}f_{n-4}f_{n-3}f_{n-3}f_{n-4}, \\ f_{n-2}f_{n-1} &= f_{n-3}f_{n-4}f_{n-2}f_{n-3} = f_{n-3}f_{n-4}f_{n-3}f_{n-4}f_{n-3}; \end{aligned}$$

luego las cadenas tienen un prefijo común de longitud $F_{n-3} + F_{n-4} + F_{n-3}$. Por hipótesis de inducción $f_{n-3}f_{n-4}$ y $f_{n-4}f_{n-3}$ tienen un prefijo común de longitud $F_{n-2} - 2$. Por lo tanto las cadenas tienen un prefijo común de longitud

$$2F_{n-3} + F_{n-4} + F_{n-2} - 2 = F_{n-2} + F_{n-1} - 2 = F_n - 2. \quad \square$$

Corolario 4.4. *Si $\Phi : \{0, 1\}^* \rightarrow \{0, 1\}^*$ se define como $\Phi(a_1 a_2 \cdots a_n) = a_1 a_2 \cdots a_{n-2}$ ($n \geq 2$), entonces se tiene que:*

$$1. \quad \Phi(f_{n-1}f_{n-2}) = \Phi(f_{n-2}f_{n-1}), (n \geq 3).$$

$$2. \Phi(f_n) = f_{n-2}\Phi(f_{n-1}), (n \geq 3).$$

Proposición 4.5. $\Phi(f_n)$ es palíndromo para todo $n \geq 2$ (i.e. f_n es palíndromo salvo por los dos últimos símbolos).

Demostración. El argumento es por inducción sobre n . Si $n = 2$, entonces $\Phi(f_2) = \lambda$, la cual es palíndromo. Supongamos que se tiene para todo entero menor o igual que n . Como $f_{n+1} = f_n f_{n-1}$ y $|f_{n-1}| \geq 2$, entonces $\Phi(f_{n+1}) = f_n \Phi(f_{n-1})$. Por lo tanto,

$$\begin{aligned} \Phi(f_{n+1})^R &= (f_n \Phi(f_{n-1}))^R \\ &= \Phi(f_{n-1})^R f_n^R \\ &= \Phi(f_{n-1}) f_n^R. \end{aligned}$$

Si n es par, entonces por la Proposición 4.2 se tiene que $f_n = \Phi(f_n)01$, luego

$$\begin{aligned} \Phi(f_{n+1})^R &= \Phi(f_{n-1})(\Phi(f_n)01)^R \\ &= \Phi(f_{n-1})10\Phi(f_n)^R \\ &= f_{n-1}\Phi(f_n) \\ &= \Phi(f_{n+1}); \end{aligned}$$

la última igualdad se obtiene por el Corolario 4.4. Para n impar el argumento es análogo. \checkmark

Corolario 4.6. Sea $f_n = \Phi(f_n)ab$. Entonces la cadena $p_n = ba\Phi(f_n)ab$ es un palíndromo.

Teorema 4.7. Para toda cadena finita f_n $n \geq 6$, se tiene que

$$f_n = f_{n-3}f_{n-3}f_{n-6}l_{n-3}l_{n-3},$$

donde $l_n = \Phi(f_n)ba$ (i.e., l_n permuta los dos últimos símbolos de f_n).

Demostración. Por definición de f_n se tiene que

$$\begin{aligned} f_n &= f_{n-1}f_{n-2} \\ &= (f_{n-2}f_{n-3})(f_{n-3}f_{n-4}) \\ &= (f_{n-3}f_{n-4})(f_{n-4}f_{n-5})f_{n-3}f_{n-4} \\ &= f_{n-3}f_{n-4}(f_{n-5}f_{n-6})f_{n-5}(f_{n-4}f_{n-5})f_{n-4} \\ &= f_{n-3}(f_{n-4}f_{n-5})f_{n-6}(f_{n-5}f_{n-4})(f_{n-5}f_{n-4}) \\ &= f_{n-3}f_{n-3}f_{n-6}t_{n-3}t_{n-3}. \end{aligned} \quad \checkmark$$

Corolario 4.8. $\Phi(f_n) = \Phi(f_{n-3})ab\Phi(f_{n-3})p_{n-6}\Phi(f_{n-3})ba\Phi(f_{n-3})$, $n \geq 6$.

Demostración. Por el teorema anterior se tiene que

$$\begin{aligned} f_n &= f_{n-3}f_{n-3}f_{n-6}l_{n-3}l_{n-3} \\ &= (\Phi(f_{n-3})ab)(\Phi(f_{n-3})ab)f_{n-6}(\Phi(f_{n-3})ba)(\Phi(f_{n-3})ba) \\ &= \Phi(f_{n-3})ab\Phi(f_{n-3})(abf_{n-6})\Phi(f_{n-3})ba\Phi(f_{n-3})ba \\ &= \Phi(f_{n-3})ab\Phi(f_{n-3})p_{n-6}p_{n-3}bap_{n-3}ba, \end{aligned}$$

así que $\Phi(f_n) = \Phi(f_{n-3})ab\Phi(f_{n-3})p_{n-6}\Phi(f_{n-3})ba\Phi(f_{n-3})$. \checkmark

Estos dos últimos resultados son importantes, ya que permiten descomponer f_n en términos de cadenas de Fibonacci de longitud menor, lo cual implica que la curva fractal de Fibonacci tiene características de autosemejanza (ver Sección 5); también es autosimilar, en el sentido de que contiene copias de sí misma, inmersas dentro de sí; para verlo, separemos las parejas 01 y cada una la reemplazamos por 0; luego reemplazamos los 0 no separados por 1:

01 0 01 01 0 01 0 01 01 0 01 01 0 ...
 0 1 0 0 1 0 1 0 0 1 0 0 1 ...

5. Representación gráfica

En esta sección vamos a implementar el lenguaje LOGO utilizado en los L-Sistemas, para interpretar gráficamente la cadena de Fibonacci. El lenguaje LOGO o Lenguaje de la Tortuga fue introducido en 1967 por W. Feurzeig y S. Papert, y su nombre deriva del griego *logos* (palabra) haciendo énfasis en que este lenguaje procesa palabras. M. Aono, T. Kunii y A. Smith en el año 1984 lo utilizaron para crear modelos de crecimiento de plantas y árboles (ver [18]). Casi al mismo tiempo, G. Siromoney y K. Subramanian mostraban que los L-sistemas podían ser interpretados para la generación de algunas curvas fractales [21]; así mismo el trabajo de P. Prusinkiewicz (1986) aborda la representación gráfica de los L-sistemas (ver [19]).

Para convertir una cadena de símbolos en una curva hay que recorrerla de una manera particular. Para ello, a cada uno de los símbolos de la cadena se le asigna una regla de dibujo que será interpretada por una tortuga hipotética que irá recorriendo el plano de un lado a otro, realizando una determinada acción, la cual dependerá del símbolo que lea (para mayor detalle ver [1] o [18]).

La tortuga responde a los comandos representados por los símbolos que aparecen en la Tabla 3.

Símbolo	Función
F	Da un paso hacia adelante dibujando una línea de longitud d . Es decir, el estado de la tortuga cambia a (x', y', θ) , donde $x' = x + d \cos \theta$ y $y' = y + d \sin \theta$. Por lo tanto, la tortuga dibuja un segmento entre los puntos (x, y) y (x', y') .
+	Gira en sentido antihorario un ángulo δ ; por lo tanto, el siguiente estado de la tortuga es $(x, y, \theta + \delta)$.
-	Gira en sentido horario un ángulo δ ; por lo tanto, el siguiente estado de la tortuga es $(x, y, \theta - \delta)$.

Cuadro 3. Comandos de LOGO.

El código del Cuadro 4 permite interpretar una cadena de símbolos basados en el lenguaje LOGO y será utilizado en la generación de las gráficas de las cadenas f_n .

Demostración. Se tiene por la Proposición 4.1, ya que no se permiten subcadenas de la forma 110 o 011. ☑

Proposición 5.4. *El número de giros en la gráfica n -ésima de Fibonacci \mathcal{F}_n es el número de Fibonacci F_{n-1} .*

La anterior propiedad es equivalente a la siguiente proposición, ya que por cada 0 se obtiene un giro en la curva.

Proposición 5.5. $|f_n|_0 = F_{n-1}$ ($n \geq 1$), es decir la cantidad de ceros en f_n es F_{n-1} .

Demostración. El argumento es por inducción sobre n . Para $n = 1$, $|f_1|_0 = |0|_0 = 1 = F_0$. Además, $|f_{n+1}|_0 = |f_n f_{n-1}|_0 = |f_n|_0 + |f_{n-1}|_0 = F_{n-1} + F_{n-2} = F_n$. ☑

En la Figura 1 se verifica que el número de giros es $F_9 = 55$ (número de esquinas).

Teorema 5.6. *La curva \mathcal{F}_n es similar a la curva \mathcal{F}_{n-3} (autosimilaridad).*

Demostración. Vamos a utilizar el homomorfismo σ^3 , donde σ es el homomorfismo de Fibonacci (ver Definición 3.2). Es claro que $\sigma^3(f_{n-3}) = f_n$. Ahora veamos que σ^3 preserva la regla de dibujo par-impar. En efecto, como $\sigma^3(0) = 01001$ y $\sigma^3(1) = 010$, entonces si $|w|$ es par se tiene que $\sigma^3(w)$ es concatenación de un número par de cadenas de longitud impar, luego $\sigma^3(w)$ es par. Análogamente, si $|w|$ es impar entonces $\sigma^3(w)$ es impar. Así, σ^3 preserva la paridad, luego cualquier subcadena en la cadena de Fibonacci preserva la paridad de su posición.

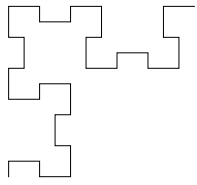
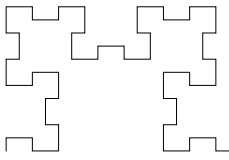
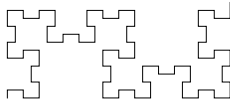
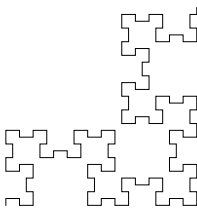
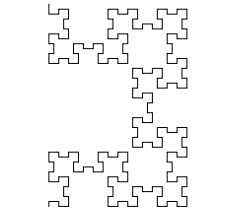
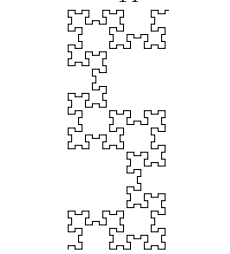
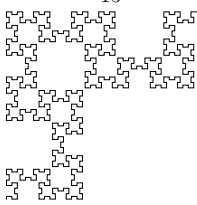
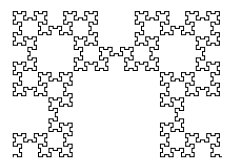
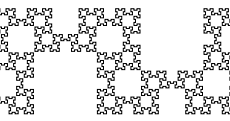
Por otra parte, definimos $a(w)$ como el ángulo en el que queda la tortuga después de leer toda la cadena w . Así, $a(00) = 0$, $a(01) = -90^\circ$ y $a(10) = 90^\circ$. Luego $a(\sigma^3(00)) = a(010010101) = 0^\circ$, $a(\sigma^3(01)) = a(01001010) = 90^\circ$ y $a(\sigma^3(10)) = a(01001001) = -90^\circ$, así que $a(w) = -a(\sigma^3(w))$, es decir σ^3 invierte el ángulo resultante. Por lo tanto la imagen de una curva por σ^3 es la simétrica de la curva dada por una simetría axial. ☑

El Teorema 5.6 permite clasificar las curvas de Fibonacci en tres clases según la relación de equivalencia módulo 3, i.e., si n es de la forma $3k$, $3k + 1$ o $3k + 2$, (ver Cuadro 5). Las gráficas son generadas con el comando del Cuadro 10 variando el número de la iteración del homomorfismo.

Teorema 5.7. *La curva \mathcal{F}_n es simétrica. Específicamente:*

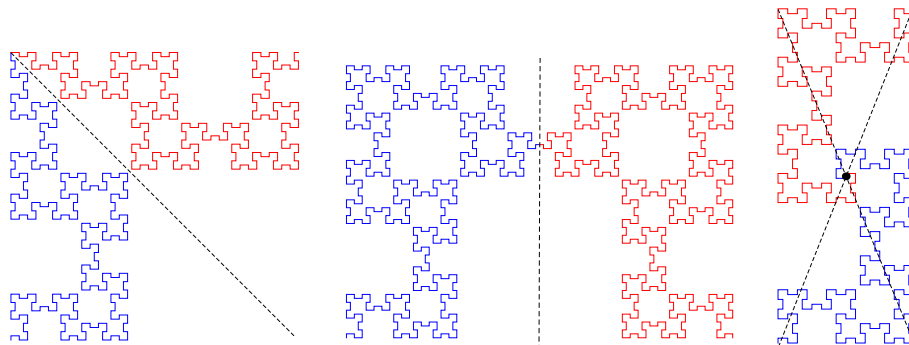
- *La curva \mathcal{F}_{3k} tiene una simetría axial diagonal.*
- *La curva \mathcal{F}_{3k+1} tiene una simetría axial ortogonal.*
- *La curva \mathcal{F}_{3k+2} tiene una simetría central.*

En el Cuadro 6 se muestran las diferentes simetrías.

$3k$	$3k + 1$	$3k + 2$
\mathcal{F}_9 	\mathcal{F}_{10} 	\mathcal{F}_{11} 
\mathcal{F}_{12} 	\mathcal{F}_{13} 	\mathcal{F}_{14} 
\mathcal{F}_{15} 	\mathcal{F}_{16} 	\mathcal{F}_{17} 

Cuadro 5. Clasificación de las curvas \mathcal{F}_n .

Demostración. Por las propiedades de la cadena de Fibonacci, se tiene que $f_n = \Phi(f_n)ab$ con $\Phi(f_n)$ palíndromo, (ver Proposición 4.5). Las dos últimas letras ab representan los dos últimos segmentos de \mathcal{F}_n , los cuales son despreciables cuando $n \rightarrow \infty$. Si el número de Fibonacci es par, entonces $|f_n|$ y $|p_n|$ son pares; así, si un cero está ubicado en la posición impar $2i + 1$ en p_n , entonces podemos encontrar otro cero (el simétrico) en la posición $F_n - 2 - (2i - 1)$, el cual es también impar. Análogamente si el número de Fibonacci F_n es impar. ☑

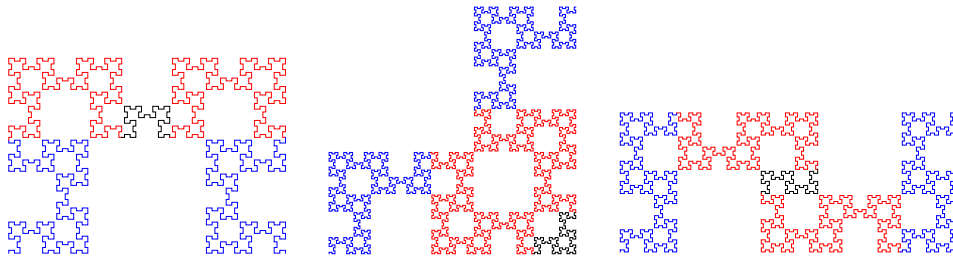


Cuadro 6. Simetrías de la Curva \mathcal{F}_n .

Teorema 5.8. La curva \mathcal{F}_n está compuesta por 5 curvas; específicamente, $\mathcal{F}_n = \mathcal{F}_{n-3}\mathcal{F}_{n-3}\mathcal{F}_{n-6}\mathcal{F}'_{n-3}\mathcal{F}'_{n-3}$, donde \mathcal{F}'_n se obtiene de aplicar la regla de dibujo a la cadena l_n .

Demostración. Se tiene por el Teorema 4.7. ☑

El teorema anterior implica que \mathcal{F}_n es la concatenación de 4 copias de \mathcal{F}_{n-3} y una copia de \mathcal{F}_{n-6} . En el Cuadro 7 se muestran las diferentes composiciones.



Cuadro 7. Composiciones de la curva \mathcal{F}_n .

Teorema 5.9. El factor de escala entre las curvas \mathcal{F}_n y \mathcal{F}_{n-3} es $1 + \sqrt{2}$; este último se llama número plateado o razón plateada.

Demostración. Por el teorema anterior \mathcal{F}_n es la concatenación de 4 copias de \mathcal{F}_{n-3} y una copia de \mathcal{F}_{n-6} . Además, por las propiedades de la cadena de Fibonacci se tiene que $\Phi(f_n) = \Phi(f_{n-3})ab\Phi(f_{n-3})p_{n-6}\Phi(f_{n-3})ba\Phi(f_{n-3})$; como $ab = 10$ o $ab = 01$, entonces las primeras dos copias (\mathcal{F}_{n-3}) son ortogonales. Análogamente se tiene para las dos últimas copias. Sea L_n la longitud desde el punto inicial de la curva \mathcal{F}_n hasta el punto final (la longitud del segmento que los une); entonces $L_n = 2L_{n-3} + L_{n-6}$ (ver Cuadro 8). Por definición, el factor escala B se define como

$$B = \frac{L_n}{L_{n-3}} = \frac{L_{n-3}}{L_{n-6}}$$

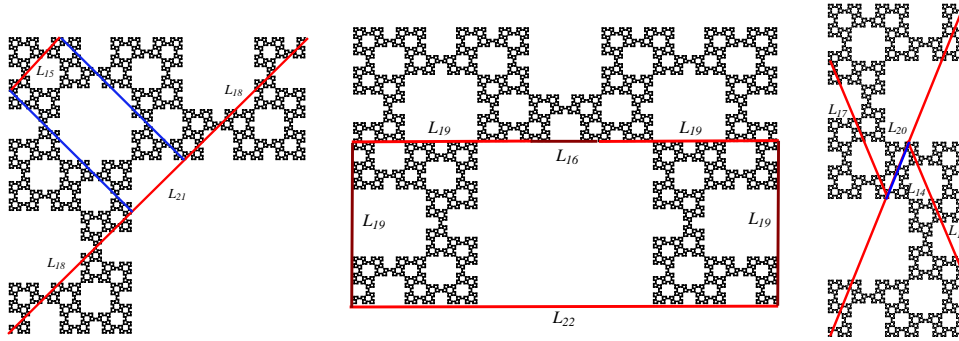
por lo tanto, $BL_{n-3} = L_n = 2L_{n-3} + L_{n-6}$; como $L_{n-6} = \frac{L_{n-3}}{B}$, entonces $BL_{n-3} = 2L_{n-3} + \frac{L_{n-3}}{B}$, luego $B = 1 + \sqrt{2}$. ☑

6. Implementación gráfica en Mathematica 8.0

Para la implementación gráfica con el *software Mathematica 8.0* de la cadena de Fibonacci, es necesario adaptar la cadena para que pueda ser interpretada por la tortuga, para lo cual se utilizará la regla de dibujo descrita en el Cuadro 9.

Para utilizar la anterior interpretación, primero generamos la cadena f de Fibonacci (comando *morfismo* del Cuadro 10; el código de este comando se muestra en el Cuadro 11; a continuación se agrupan por parejas los símbolos de la cadena (comando *Partition* del Cuadro 10), y a partir de lo anterior se aplica el siguiente homomorfismo a f :

$$10 \rightarrow FF+, \quad 01 \rightarrow F-F, \quad 00 \rightarrow F-F+.$$



Cuadro 8. Longitud de la Curva \mathcal{F}_n .

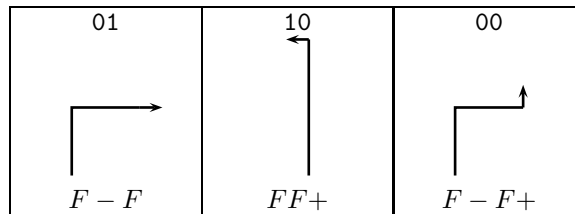
Símbolo	Función
10	Traza una línea de longitud $2d$ y gira a la izquierda.
01	Traza una línea de longitud d , gira a la derecha y traza una línea de longitud d .
00	Traza una línea de longitud d , gira a la derecha, traza una línea de longitud d y gira a la izquierda.

Cuadro 9. Nueva regla de dibujo.

La cadena resultante solo contiene ahora los símbolos F , $+$ y $-$, los cuales son interpretados por la tortuga con el comando LMove (ver Cuadro 4).

```

----- Mathematica 8.0 -----
1 morfismo[{0}, {1 -> 0, 0 -> {0, 1}}, 9]
2 Partition[%, 2] /. {{1, 0} -> "FF+", {0, 1} -> "F-F", {0, 0} -> "F-F+"} // StringJoin;
3 LMove[%, 90 Degree, {{0, 0}, 90 Degree}];
4 Graphics[Line[%]]
    
```



Cuadro 10. Regla de dibujo para la cadena de Fibonacci.

En la Sección 8.7 mostramos otra forma de graficar la curva f_n .

6.3. Algunas variaciones

A continuación mostramos algunas variaciones de la cadena de Fibonacci, cuya curva asociada en algunos casos tiene como atractor la curva fractal de Fibonacci. Estas variaciones se obtienen a partir de una nueva cadena de símbolos, denominada *cadena densa de Fibonacci* (introducida en [15]).

```

1 morfismo[\[CapitalSigma]_, \[Mu]_, n_] := Block[{\[CapitalDelta] = \[CapitalSigma]},
2 Do[\[CapitalDelta] = Flatten[\[CapitalDelta] /. \[Mu]], {n}]; Return[\[CapitalDelta]];

```

Cuadro 11. Código para el comando morfismo.

Definición 6.1. La *cadena densa de Fibonacci* \hat{f} es la que resulta de aplicar el homomorfismo

$$\eta(00) = 0, \quad \eta(01) = 1, \quad \eta(10) = 2$$

a la cadena de Fibonacci f . Así, $\hat{f} = 102210221102110211022102211021 \dots$.

Si se aplica un homomorfismo que preserve o invierta los ángulos a la cadena densa de Fibonacci, se obtienen curvas cuyo límite es la cadena fractal de Fibonacci (ver [15]). En la Tabla 13 se muestran algunas variaciones de \hat{f} , utilizando diferentes homomorfismos. Estas gráficas son generadas utilizando el código que aparece en el Cuadro 12, donde la línea 1 genera la cadena f , la línea 2 genera \hat{f} y finalmente la línea 3 produce la cadena generada por el homomorfismo que preserva el ángulo. Esta línea se puede cambiar por un homomorfismo que preserve ángulos, y así obtener nuevas curvas.

```

1 morfismo[{0}, {1 -> 0, 0 -> {0, 1}}, 15];
2 Partition[%, 2] /. {{1, 0} -> "2", {0, 1} -> "1", {0, 0} -> "0"} // StringJoin;
3 StringReplace[%, {"1" -> "02", "0" -> "21", "2" -> "10"}];
4 StringReplace[%, {"0" -> "F", "1" -> "F-", "2" -> "F+"}];
5 LMove[%, 90 Degree, {{0, 0}, 90 Degree}];
6 Graphics[Line[%]]

```

Cuadro 12. Código para generar curvas cuyo atractor es la curva de Fibonacci.

Las posibilidades de obtener gráficas se incrementan si permitimos ahora que el ángulo varíe ($N[90 \text{ Degree}]$). En la Tabla 14 aparecen algunos ejemplos.

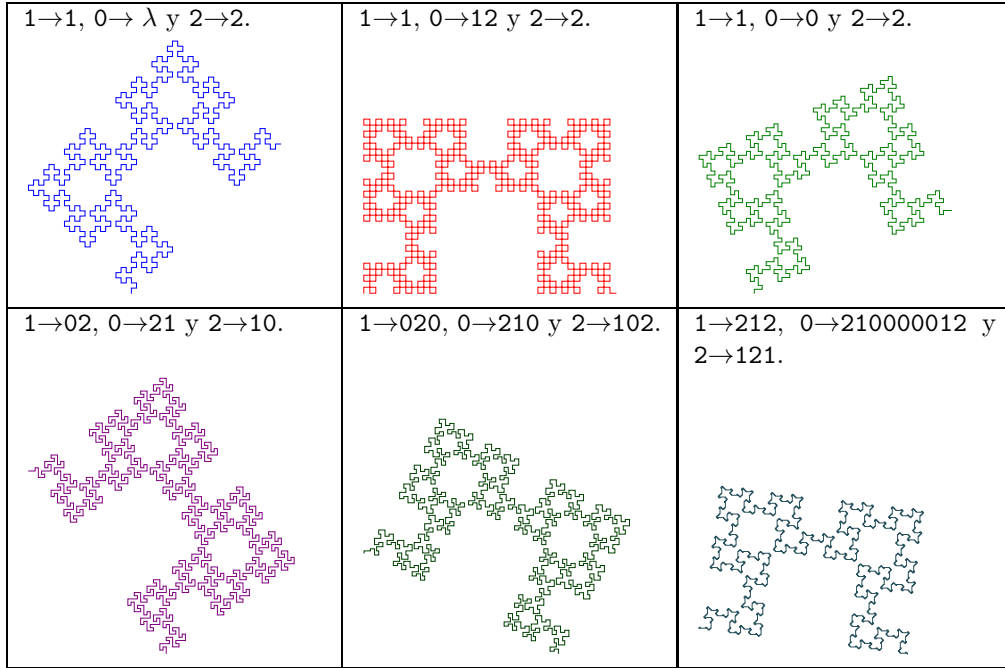
6.4. Otras cadenas

El procedimiento presentado en las secciones anteriores para generar curvas a partir de cadenas de símbolos se puede aplicar en otras cadenas, como se muestra en el siguiente ejemplo y en la próxima sección.

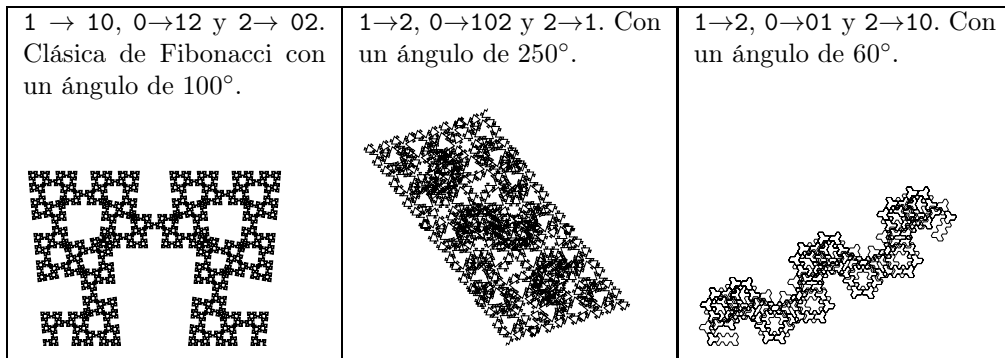
Ejemplo 6.2. Consideremos la cadena de *Baum-Sweet* (ver, e.g., [3, p. 156]) $\mathbf{b} = (b_n)_{n \geq 1} = 1101100101001001 \dots$, donde $b_n = 1$ si la representación binaria de n no contiene bloques de consecutivos 0 de longitud impar y $b_n = 0$ en caso contrario. Esta cadena se puede generar a partir del siguiente homomorfismo:

$$\gamma(00) = 0000, \quad \gamma(01) = 1001, \quad \gamma(10) = 0100, \quad \gamma(11) = 1101,$$

donde $\gamma^n(11) = \mathbf{b}$ cuando $n \rightarrow \infty$.



Cuadro 13. Curvas cuyo atractor es la curva Fractal de Fibonacci.



Cuadro 14. Curvas generadas por variación del ángulo.

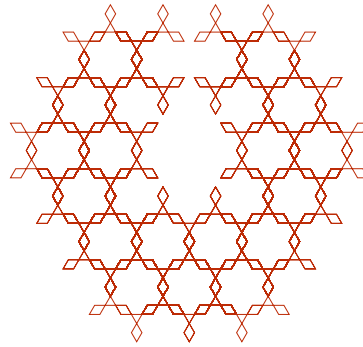
Si a la cadena de Baum-Sweet le aplicamos el homomorfismo

$$\nu(0) = +F + F+, \quad \nu(1) = -F-,$$

se obtiene la curva que aparece en el Cuadro 15 (10 iteraciones).

7. Cadena infinita de Thue-Morse

En esta sección estudiamos la cadena binaria conocida como *Thue-Morse*, la cual es sin duda la palabra más interesante después de la de Fibonacci (por esta razón las presentamos en conjunto). Esta cadena infinita fue introducida por el matemático noruego



Cuadro 15. Curva generada por el homomorfismo ν aplicado a la cadena de Baum-Sweet.

Axel Thue (1863–1922) en 1906, cuando estudiaba repeticiones dentro de cadenas, y redescubierta por Marston Morse (1892–1977) en 1921 [16]. Ella constituye una definición ubicua en la literatura [2], y posee diferentes maneras equivalentes de construirse, entre las cuales presentamos las siguientes tres.

Definición 7.1. Definimos la cadena de *Thue-Morse*² como

$$\mathbf{t} = (t_n)_{n \geq 0} = 01101001100 \dots ,$$

donde

$$t_n = \begin{cases} 0, & \text{si la cantidad de 1's en la expansión en base 2 de } n \text{ es par,} \\ 1, & \text{si la cantidad de 1's en la expansión en base 2 de } n \text{ es impar.} \end{cases}$$

Definición 7.2. Dados $\Sigma = \Delta = \{0, 1\}$, se define el *homomorfismo de Thue-Morse* como $\mu(0) = 01$ y $\mu(1) = 10$. Entonces

$$\mathbf{t} := \lim_{n \rightarrow \infty} \mu^n(0) = 011010011001011010010110011010011001011001101001 \dots$$

Se puede probar que \mathbf{t} es la única palabra que comienza con 0 y es punto fijo para μ , es decir, $\mu(\mathbf{t}) = \mathbf{t}$.

Por otra parte, si iteramos en 1 obtenemos la palabra infinita $\bar{\mathbf{t}}$, que es el otro punto fijo (la cual es Thue-Morse cambiando el 0 por el 1 y viceversa):

$$\bar{\mathbf{t}} := \lim_{n \rightarrow \infty} \mu^n(1) = 1001011001101001011010011001011001101001100101101001 \dots$$

Definición 7.3. Otra manera de producir la cadena de Thue-Morse es comenzar con 0 e iterar el siguiente proceso: tome la cadena actual y añada su complemento. (Por complemento entendemos reemplazar 0 por 1 y 1 por 0.) El proceso corre de manera

²Para un estudio detallado de la cadena de Thue-Morse ver [3] y [2].

simple:

0
 01
 0110
 01101001
 0110100110010110
 ...

Después de la generación inicial, cada palabra t_n generada tiene 2^n veces el 0 y 2^n veces el 1. Nótese que t_{2n} es palíndromo. Si utilizamos la Definición 7.2 obtenemos que

$$t_{n+1} = \mu^n(0)\mu^n(1).$$

Por la construcción anterior es fácil ver que la cadena de Thue-Morse es libre de cubos, es decir no contiene subcadenas de la forma 000, 111, 010101, 010010010, o más generalmente, w^3 .

Por supuesto que el código para generar t_n también es simple:

```

1 Nest[StringReplace[#, {"0" -> "01", "1" -> "10"}] &, "0", n]

```

Teorema 7.4. $\mu^n(0) = t_n$ para todo $n \geq 0$.

Demostración. Por inducción sobre n . El caso $n = 0$ es inmediato. Supongamos entonces que es verdad para n , y demostremos que lo es para $n + 1$. Tenemos que

$$\mu^{n+1}(0) = \mu^n(\mu(0)) = \mu^n(01) = \mu^n(0)\mu^n(1) = t_{n+1}. \quad \checkmark$$

La cadena de Thue-Morse es autosimilar en el buen sentido se los fractales; es decir, ella contiene infinitas copias de sí misma. Para observarlo, basta con cancelar todos los términos pares t_{2n} de t , y nuevamente obtenemos la palabra original t :

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

7.5. Interpretación geométrica de la cadena de Thue-Morse

Si visualizamos el 0 como un cuadrado blanco y el 1 como un cuadrado negro, la cadena de Thue-Morse es representada gráficamente como en la Figura 2 (por supuesto, esta interpretación es válida para cualquier palabra, pero en el caso de Thue-Morse es especialmente bella en razón de la ‘simetría’ de las palabras):



Figura 2. Palabra $\mu^4(0) = 01101001100110$.

Esta construcción puede ser extendida a dos dimensiones si en cada caso añadimos el complemento, tanto horizontal como vertical (la Figura 3 muestra las primera cuatro

iteraciones), y obtenemos el plano de Thue-Morse, el cual es fractal. Y, a pesar de su apariencia de simetría y regularidad, no hay repeticiones. Es decir, ninguna porción finita del plano puede ser tomada como una baldosa para reproducir o cubrir todo el plano [9].

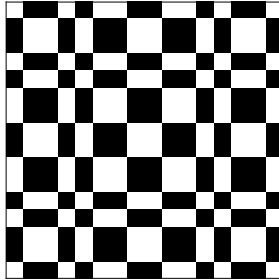


Figura 3. Palabra $\mu^4(0)$ bidimensional.

7.6. Interpretación fractal de la cadena de Thue-Morse

Por supuesto, los símbolos 0 y 1 utilizados en la definición de t no tienen nada de especial; es decir, podemos escoger cualquier otro alfabeto de dos letras $\Sigma = \{F, +\}$. En este caso el símbolo F representa un movimiento hacia adelante y $+$ una rotación para la tortuga en un ángulo $\delta = 120^\circ$. Esta construcción es el Ejemplo 7.5.

Ejemplo 7.5. En [14] los autores muestran una estrecha relación entre la cadena de Thue-Morse y la curva de Koch (el clásico fractal introducido por Helge von Koch in 1906, el cual es una curva de longitud infinita conteniendo una área finita [10]); sin embargo, esta relación ya había sido observada por F. M.Dekking (ver, e.g., [5], [6] y [7]) y Deshouillers, J.-M. (ver, e.g., [8]). Para mayores detalles sobre este aspecto ver [4].

La curva de Koch es obtenida aplicando el siguiente homomorfismo:

$$\tau(0) = F, \quad \tau(1) = -.$$

El código en Mathematica 8.0 se muestra en la Figura 4 junto con la gráfica para el caso de 10 iteraciones; en la Figura 5 se muestra \bar{t} , y en 6 aparecen algunas iteraciones.

```

1 Nest[StringReplace[#, {"F" -> "F-", "-> "-F"}] &, "F", 10];
2 LMove[%, 120 Degree, {{0, 0}, 120 Degree}]; Graphics[{Line[%]}]
    
```



Figura 4. Curva de Koch generada por la iteración del homomorfismo $\mu^{10}(0)$.

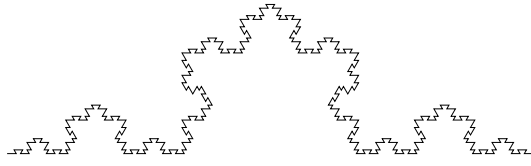


Figura 5. Curva de Koch generada por la iteración del homomorfismo $\mu^{10}(1)$.

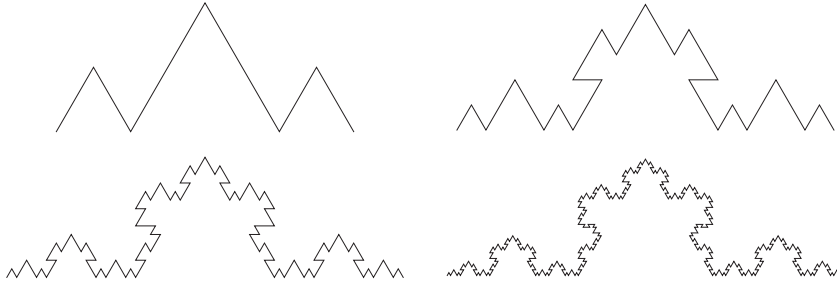


Figura 6. Palabras t_4, t_6, t_8 y t_{10} .

8. Cadenas de Sturm

Las cadenas de Sturm admiten diferentes definiciones que resultan equivalentes, y pueden ser descritas explícitamente de manera aritmética. La manera corriente de definir las palabras de Sturm es por medio de la función de complejidad P .

Definición 8.1. Dada una cadena infinita $\mathbf{w} = w_1w_2w_3\cdots$ sobre un alfabeto finito, definimos la función $P_w : \mathbb{N} \rightarrow \mathbb{N}$ de la manera siguiente: notamos con $L_n(\mathbf{w})$ el conjunto de todas las subcadenas de \mathbf{w} de longitud n , es decir,

$$L_n(\mathbf{w}) = \{w_jw_{j+1}\cdots w_{j+n-1} : j \geq 1\}.$$

La *función de complejidad* $P_{\mathbf{w}}(n)$ es definida como la cardinalidad de $L_n(\mathbf{w})$ (el número de diferentes subcadenas de longitud n que tiene \mathbf{w}).

Definición 8.2. Una cadena infinita \mathbf{w} es una cadena de Sturm si $P(\mathbf{w}, n) = n + 1$ para todo entero $n \geq 0$ (la complejidad es mínima con respecto a todas las cadenas que no son a la larga periódicas³).

Como $P(w, 1) = 2$, entonces las palabras de Sturm están construidas sobre alfabetos de solo dos letras. Claramente, $P(w, 0) = 1$ para toda w .

Ejemplo 8.3. La cadena infinita de Fibonacci \mathbf{f} es de Sturm [12], es decir se tiene que $P(\mathbf{f}, n) = n + 1$ para todo n (ver Cuadro 16).

El código del Cuadro 17 permite generar las diferentes subcadenas de una cadena finita.

³Un resultado célebre de M. Morse y G. Hedlund afirma que una sucesión es a la larga periódica si $P(n) \leq n$ para algún n ([17]). Por tanto, entre todas las palabras no periódicas, las de Sturm son aquellas que tienen la menor complejidad posible. También acuñaron en 1940 el término Sturm para referirse a estas palabras, en honor al matemático Jacques Ch. F. Sturm.

$P(f, 0) = 1$	$P(f, 1) = 2$	$P(f, 2) = 3$	$P(f, 3) = 4$	$P(f, 4) = 5$
λ	0, 1	00, 01, 10	010, 001, 100, 101	0100, 1001, 0010, 0101, 1010

Cuadro 16. Función de Complejidad para f .

```

----- Mathematica 8.0 -----
1 StringPartition[string_, n_] := Table[StringTake[string, {i, i+n-1}],
2   {i, 1, StringLength[string] - (n - 1)}];
3 subcadenas[string_, n_] := Intersection[StringPartition[string, n]]
4 subcadenas["01001010010010010101010", 5] % Ejemplo n=5
5 { 00100, 00101, 01001, 01010, 10010, 10100 }
```

Cuadro 17. Código para generar las diferentes subcadenas de longitud n .

Ejemplo 8.4. La cadena t de Thue-Morse no es una cadena de Sturm, ya que tiene por ejemplo $P(t, 2) = 4$.

Como una propiedad interesante tenemos que la Proposición 4.5 se tiene en toda cadena de Sturm [12].

Una manera aritmética equivalente de definir una cadena de Sturm es la siguiente (existen diferentes maneras equivalentes de definir estas cadenas (ver, e.g., [3] o [12])). Dada la cadena infinita $w = w_1 w_2 w_3 \dots$, decimos que w es de Sturm si existen $\alpha, \beta \in \mathbb{R}$ con $0 \leq \alpha \leq 1$ irracional, y tales que

$$w_n = \lfloor \alpha(n + 1) + \beta \rfloor - \lfloor \alpha n + \beta \rfloor.$$

8.7. Cadenas características y curvas fractales

Como un caso particular de las cadenas de Sturm (el caso en que $\beta = 0$), se tienen las cadenas características, las cuales permiten generar nuevas curvas.

Definición 8.5. Sea α un número real tal que $0 < \alpha < 1$. Para $n \geq 1$, definimos:

$$w_\alpha(n) := \lfloor (n + 1)\alpha \rfloor - \lfloor n\alpha \rfloor$$

y

$$w(\alpha) := w_\alpha(1)w_\alpha(2)w_\alpha(3) \dots$$

Entonces $w(\alpha)$ se llama cadena característica de pendiente α .

Ejemplo 8.6. La cadena f de Fibonacci es la palabra característica de pendiente $\frac{3-\sqrt{5}}{2}$. Exactamente, $f = w\left(\frac{1}{\phi^2}\right)$, donde $\phi = \frac{1+\sqrt{5}}{2}$ es la razón áurea.

$$w\left(\frac{1}{\phi^2}\right) = 010010100100101001010010010100100100101001010010010010 \dots = f.$$

Explícitamente se tiene que el símbolo n -ésimo es 0 (respectivamente 1) si $\lfloor (n + 1)\tau \rfloor - \lfloor n\tau \rfloor = 0$ (respectivamente 1).

```

1 Sturmian[\[Alpha]_, n_Integer] :=Table[Floor[\[Alpha] (i + 1)] - Floor[\[Alpha] i];
2
3 DenseMove[z_List, \[Delta]_, pos_List]:= Block[{x, y, \[Theta], moves},{x, y}=pos[[1]];
4 \[Theta] = pos[[2]];moves = {{x, y}}; Map[({x, y} += {Cos[\[Theta]], Sin[\[Theta]]});
5 Switch[#, 1, \[Theta] += \[Delta], 2, \[Theta] -= \[Delta]];
6 AppendTo[moves, {x, y}] &, z];Return[moves];] ;
7
8 Map[2 #[[1]] + #[[2]] - 1 &, Partition[Sturmian[\[Pi]-3, 4000], 2]];
9 DenseMove[%, N[100.4 Degree], {{0, 0}, N[30 Degree]}];
10 g1 = Graphics[Line[%]]

```

Cuadro 18. Código para generar la cadena característica.

El código del Cuadro 2 utiliza la anterior definición para generar f_n , y el código del Cuadro 18 permite generar de manera general la cadena característica $w(\alpha)$ y su curva asociada.

Utilizando las reglas de dibujo y diferentes cadenas características se pueden generar nuevas curvas, como se muestra en la Figura 7. El uso de las cadenas características para generar curvas, es un posible camino, no muy explorado en la literatura, el cual es un trabajo para futuro para desarrollar con detalle.

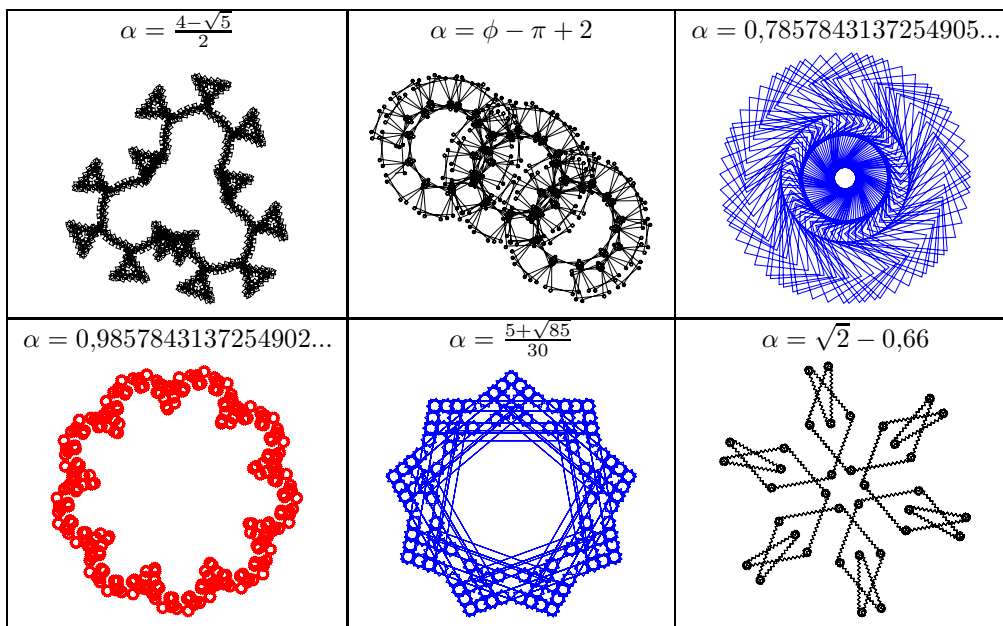


Figura 7. Ejemplos de curvas generadas por cadenas características.

Una variación interesante en el código del Cuadro 18 es cambiar el comando `Line` por el comando `Polygon`; en la Figura 8 se muestra un ejemplo. Así mismo, en la Figura 9 se muestra un *zoom* de una de las cadenas características.

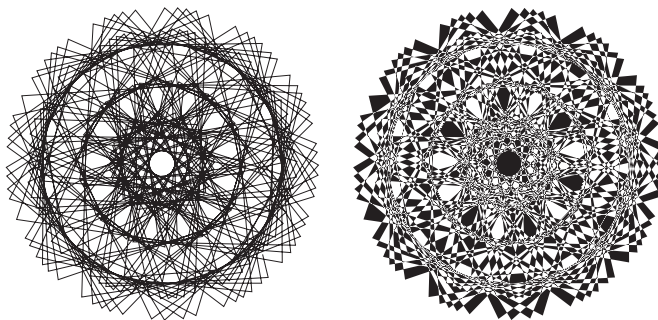


Figura 8. Variación cadena característica con $\alpha = \pi - 3$.

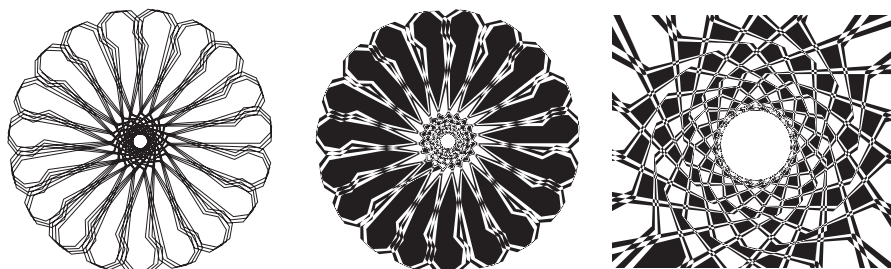


Figura 9. Zoom a una cadena característica con $\alpha = 0,54761\dots$

9. Agradecimientos

El primer autor está parcialmente financiado por la Universidad Sergio Arboleda (proyecto USA-2011-0059). Los autores agradecen los valiosos comentarios y sugerencias de los árbitros, los cuales enriquecieron y mejoraron este trabajo.

Agradecemos al profesor Borut Jurcic-Zlobec de la Universidad de Ljubljana por sus valiosas sugerencias en la implementación del código en Mathematica.

Referencias

- [1] Abelson H., diSessa A.A., *Turtle geometry*, MIT Press Series in Artificial Intelligence, MIT Press, Massachusetts, 1981.
- [2] Allouche J.-P., Shallit J., "The ubiquitous Prouhet-Thue-Morse sequence", En Ding C., Hellesteth T., Niederreiter H., editors, *Sequences and their Applications*, Proceedings of SETA'98, Springer Verlag (1999), 1–16.
- [3] Allouche J.-P. Shallit J., *Automatic Sequences*, Cambridge University Press, Cambridge, 2003.
- [4] Allouche J.-P. Skordev G., "Von Koch and Thue-Morse revisited", *Fractals* 15 (2007), no. 4, 405–409.
- [5] Dekking F.M. "Recurrent sets", *Adv. in Math.* 44, (1982), no. 1, 78–104.

- [6] Dekking F.M., “Replicating superfigures and endomorphisms of free groups”, *J. Combin. Theory Ser. A* 32 (1982), 315–320.
- [7] Dekking F.M., Mendès M., “Uniform distribution modulo one: a geometrical viewpoint”, *J. Reine Angew. Math.* 329 (1981), 143–153.
- [8] Deshouillers J.-M., “Geometric aspects of Weyl sums”, Elementary and analytic theory of numbers (Warsaw, 1982), *Banach Center Publ.* 17 (1985), 75–82.
- [9] Griswold R.E., “The Morse-Thue Sequence”, Dep. of Computer Science, The University of Arizona. <http://www.cs.arizona.edu/patterns/weaving/>
- [10] Von Koch H., “Une méthode géométrique élémentaire pour l’étude de certaines questions de la théorie des courbes planes”, *Acta Math.* 30 (1906), no. 1, 145–174.
- [11] Lothaire M., *Combinatorics on Words*, Cambridge University Press, Cambridge, 1983.
- [12] Lothaire M., “Algebraic Combinatorics on Words”, *Encyclopedia of Mathematics and its Applications*, Cambridge University Press, Cambridge, 2002.
- [13] Lothaire M., “Applied Combinatorics on Words”, *Encyclopedia of Mathematics and its Applications*, Cambridge University Press, Cambridge, 2005.
- [14] Ma J., Holdener J. “When thue Morse meets Koch”, *Fractals* 13 (2005), no. 3, 191–206.
- [15] Monnerot A., “The Fibonacci Word Fractal”, preprint (2009).
<http://hal.archives-ouvertes.fr/hal-00367972/fr/>
- [16] Morse M., “Recurrent geodesics on a surface of negative curvature”, *Transactions Amer. Math. Soc.* 22 (1921), no. 1, 84–100.
- [17] Morse M., Hedlund G., “Symbolic Dynamics II: Sturmian Sequences”, *Amer. J. Math.* 62 (1940), 1–42.
- [18] Prusinkiewicz P., Lindenmayer A., *The algorithmic beauty of plants*, Springer-Verlag, New York, 2004. <http://algorithmicbotany.org/papers/abop/abop.pdf>.
- [19] Prusinkiewicz P., “Graphical applications of L-systems”, *Proceedings of Graphics Interface* (ed. Wein M. and Kidd E.M.), Canadian Information Processing Society (1986), 247–253.
- [20] Schützenberger M-P., “Une théorie algébrique du codage,” *In Séminaire Dubreil-Pisot* 1955–56, Exposé No. 15, (1955).
- [21] Siromoney R., Subramanian K., “Space-filling curves and infinite graphs”, *Graph grammars and their application to computer science* (ed. Ehrig H., Nagl M. and Rozenberg G.), Second International Workshop, Lecture Notes in Computer Science 153, Springer-Verlag, Berlin (1983), 380–391.