RELACIÓN ENTRE SECUENCIAS AUTOMÁTICAS Y HOMOMORFISMOS UNIFORMES DESDE EL TEOREMA DE COBHAM

GABRIEL FERNANDO MONCADA SANTOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE CIENCIAS

MATEMÁTICAS

BUCARAMANGA

2023

RELACIÓN ENTRE SECUENCIAS AUTOMÁTICAS Y HOMOMORFISMOS UNIFORMES DESDE EL TEOREMA DE COBHAM

GABRIEL FERNANDO MONCADA SANTOS

Trabajo de grado para optar al titulo de Matemático

Director

CARLOS ARTURO RODRÍGUEZ PALMA

Doctor en Matemáticas

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE CIENCIAS

MATEMÁTICAS

BUCARAMANGA

2023

Dedicatoria

Dedicado a la memoria de mi nonita Isbelia, mi papá Betico, mi tío Luis, mi compañero Kennethy, mi amigo Luca y mi perrita Muñequita, quienes viven en mis recuerdos y me llenan de motivación para seguir adelante.

Agradecimientos

Infinitas gracias a mi madre, Blanca Cecilia Santos, a quien le debo todo en mi vida. Muchas gracias a todos aquellos que a pesar de las circunstancias, los tropiezos y los buenos momentos, hoy en día siguen a mi lado, apoyándome y creyendo en mí.

Tabla de Contenido

Lista de figuras

Lista de tablas

RESUMEN

TITULO: RELACIÓN ENTRE SECUENCIAS AUTOMÁTICAS Y HOMOMORFISMOS UNIFORMES DESDE EL TEOREMA DE COBHAM.[1]

AUTOR: GABRIEL FERNANDO MONCADA SANTOS.[2]

PALABRAS CLAVE: COMBINATORIA SOBRE CADENAS, AUTÓMATAS, SECUENCIAS AUTOMÁTICAS, HOMOMORFISMOS UNIFORMES, COBHAM, GEOMETRÍA FRACTAL, ARTE MATEMÁTICO .

DESCRIPCIÓN:

Este documento se interesa por estudiar la clase de secuencias infinitas sobre un alfabeto Σ que son generadas por un autómata de estados finito y que a su vez también son generadas por la iteración fija de un homomorfismo uniforme; llamadas, secuencias automáticas.

A partir del estudio de las condiciones necesarias para que una secuencia sobre un alfabeto Σ pueda ser generada de estas dos maneras y otras propiedades importantes que se desprenden de las secuencias automáticas se pretende generar un estudio experimental de las diferentes curvas y gráficas que se pueden generar con esta clase de secuencias y las reglas de dibujo análogas a las usadas en los L-Sistemas a partir de programación en `Python` con la motivación de generar arte matemático y un repaso por fractales ya conocidos.

[1] Trabajo de grado.

[2] FACULTAD DE CIENCIAS, ESCUELA DE MATEMÁTICAS.
DIRECTOR: DOCTOR CARLOS ARTURO RODRÍGUEZ PALMA.

ABSTRACT

TITLE: RELATIONSHIP BETWEEN AUTOMATIC SEQUENCES AND UNIFORM HOMOMORPHISMS FROM COBHAM'S THEOREM.[3]

AUTHOR: GABRIEL FERNANDO MONCADA SANTOS.[4]

KEYWORDS: COMBINATORICS ON WORDS, AUTOMATAS, AUTOMATIC SEQUENCES, UNIFORM HOMOMORPHISMS, COBHAM, FRACTAL GEOMETRY, MATHEMATICAL ART.

DESCRIPTION:

This document is interested in studying the class of infinite sequences over an alphabet Σ that are generated by a finite automaton and are also generated by the fixed iteration of a uniform homomorphism; calls, automatic sequences.

From the study of the necessary conditions for a sequence on an Σ alphabet to be generated in these two ways and other important properties that emerge from automatic sequences, it is intended to generate an experimental study of the different curves and graphs that they can be generated with this class of sequences and drawing rules analogous to those used in L-Systems from `Python` programming with the motivation of generating mathematical art and a review of already known fractals.

[3]Bachelor thesis.

[4]FACULTAD DE CIENCIAS, ESCUELA DE MATEMÁTICAS.
DIRECTOR: DOCTOR CARLOS ARTURO RODRÍGUEZ PALMA.

INTRODUCCIÓN

El matemático americano y científico en computación Alan Belmont Cobham, interesado en el estudio de los conjuntos de números enteros que un autómata finito puede reconocer a partir de su representación en una determinada base en el año de 1969 escribió el artículo titulado *On the base-dependence of sets of numbers recognizable by finite automata* [1] en el cual muestra que para que un conjunto de enteros no negativos sea reconocido por un autómata, el conjunto debe ser en última instancia periódica, esto es, que el conjunto con algunas excepciones finitas es la unión de un sistema de clases de equivalencia módulo algún entero positivo fijo. Más adelante, en el año 1972, en el artículo titulado *Uniform tag sequences* [2] caracterizó las secuencias automáticas y desarrolló su teoría obteniendo diferentes resultados que las relacionan con diferentes áreas de la matemática, tales como la computación y el álgebra. En el año de 2003 el matemático francés Jean-Paul Allounche y el científico computacional estadounidense Jeffrey Shallit publicaron el libro *Automatic Sequences* [3], recopilando con gran detalle diferentes propiedades, generalizaciones y teoría relacionada a las secuencias caracterizadas como automáticas. Posteriormente en el año 2022 nuevamente Jeffrey Shallit publicaría un nuevo libro titulado *The logical approach to automatic sequences: Exploring combinatorics on Words with Walnut* [4], donde complementaría y se enfocaría en dar una aproximación lógica de primer orden a estas secuencias y sus propiedades que fueron estudiadas por mucho tiempo en la combinatoria de palabras. Finalmente Shallit complementa su trabajo con la muestra de un **procedimiento de decisión** implementado en su programa de computadora llamado `Walnut` y que permite demostrar de forma sencilla muchos resultados que también son recopilados por el autor en docenas de artículos y tesis doctorales sobre secuencias que se encuentren caracterizadas como secuencias automáticas.

Los dos libros mencionados anteriormente junto con los artículos de Cobham constituyen una base de referencias sólida a la hora de proponer un estudio sobre las secuencias automáticas y de los cuales muchos resultados y definiciones son nuevamente presentados en este documento con el fin de hacer auto-contenido.

Uno de los principales artículos que motivaron la realización de este documento fue publicado en el año 2012 en la revista *Integración* de la Universidad Industrial de Santander,

el artículo *Generación de curvas fractales a partir de homomorfismos entre lenguajes [con Mathematica ®]* [5], de los matemáticos colombianos José L. Ramírez y Gustavo N. Rubiano, muestra una implementación de el uso del software Mathematica 8.0 en algunas propiedades combinatorias para las secuencias (o sucesiones), o como se verá más a delante; para las cadenas infinitas de Fibonacci y Thue-Morse.[5] A partir del estudio de estas secuencias se mostró que la construcción de estas cadenas se puede generar a partir de la iteración de un homomorfismo entre lenguajes. Es esta propiedad es una de las principales motivaciones de este documento, ya que a partir de esta relación como en el artículo mencionado, se hará uso del lenguaje `LOGO` o lenguaje de la tortuga[6], para generar gráficas de las curvas fractales asociadas a estas cadenas y así mismo recopilar nuevas propiedades a partir de las curvas fractales generadas por secuencias automáticas que también cumplen con esta propiedad.

Es importante resaltar también la motivación generada por el artículo *Artwork based on automatic sequences* [6] del matemático canadiense John Campbell, publicado en el año 2022 en la revista *Journal of Mathematics and the Arts*, dónde se muestran también una fusión interdisciplinaria entre las secuencias automáticas y el arte matemático generado a partir de ellas.

Este documento presenta una revisión bibliográfica de los diferentes artículos y libros enmarcados en diferentes campos de la matemática, como la computación, la combinatoria de palabras, el arte matemático y otras disciplinas como lo son las gramáticas y lenguajes formales que permite mostrar al lector una base teórica a la hora de definir las secuencias automáticas y parte de su entorno, y así mismo mostrar una relación entre estos diferentes campos; Para ello el documento está divido en capítulos y secciones; comenzando con un primer capítulo de preliminares que muestra las definiciones básicas de la teoría de combinatoria de cadenas, lenguajes y expresiones regulares, homomorfismos uniformes y una breve introducción de la representación en base k de números enteros positivos. En el segundo capítulo se presenta una introducción a la teoría computacional con la presentación de algunos modelos de autómatas de estados finitos. El tercer capítulo está destinado a caracterizar a las secuencias automáticas

[5]La cadena infinita de Fibonacci, lleva dicho nombre ya que su construcción es análoga a la definición recursiva de los números de Fibonacci.

[6]El lenguaje logo es un conjunto de reglas de dibujo análogas y similares a las utilizadas en los L-Sistemas introducidos por el biólogo Astrid Lindenmayer en 1986.

y a partir de estas, definir los conjuntos automáticos y finalmente mostrar la relación que existe entre secuencias automáticas y los homomorfismos uniformes. Finalmente en el cuarto y último capítulo se introduce al lector en las gramáticas generativas, para así definir a los L-Sistemas y mostrar su interpretación gráfica a partir de el lenguaje de la tortuga. Mostrando diferentes curvas programadas en `Python`[7] y que también pueden ser generadas a partir de una secuencia automática o a partir de la caracterización por medio de homomorfismos y las reglas de dibujo análogas a los L-Sistemas en el lenguaje de la tortuga, para finalizar con algunas conclusiones.

[7] `Python` es un lenguaje de alto nivel de programación interpretado, creado por el informático Guido Van Rossum en 1986, que a la fecha es uno de los más usados y útiles en áreas de la ciencia computacional e ingeniería por su lenguaje versátil y fácil de aprender [7], además de su acceso gratuito y con una licencia de código abierto, el cuál será usado en el último capítulo para visualizar las diferentes curvas generadas por las secuencias automáticas, por medio de la programación con este lenguaje y algunas de sus librerías.

1. PRELIMINARES

En este capítulo inicial se presenta los conceptos preliminares y definiciones básicas e introductorias de la combinatoria de cadenas, lenguajes regulares, homomorfismos uniformes y el sistema de numeración en base-k, los cuales permiten introducir al lector en la lectura de los siguientes capítulos. La teoría que aquí se presenta se ha tomado como referencia de los libros [3], [4], [8], [9] y [10].

1.1. Alfabetos y cadenas

Definición 1.1.1 (Alfabeto)**.** Un **alfabeto** es un conjunto finito no vacío, denotado por la letra griega Σ, cuyos elementos se denominan símbolos.

Ejemplo 1.1.2. El alfabeto del idioma castellano es el conjunto:

$$\Sigma = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{y}, \mathtt{z}, \mathtt{A}, \mathtt{B}, \ldots, \mathtt{Y}, \mathtt{Z}\}.$$

Notación 1.1.3. Dado un entero $k \geq 2$, se denota por Σ_k al alfabeto dado por el conjunto $\Sigma_k = \{0, 1, 2, \ldots, k-1\} \subset \mathbb{N}$;[1] En particular, el alfabeto $\Sigma_2 = \{0, 1\}$ es denominado el alfabeto binario.

Definición 1.1.4 (Cadena finita)**.** Sea Σ un alfabeto, entonces una **cadena** o **palabra** sobre Σ es una sucesión (o secuencia) finita de elementos elegidos del conjunto Σ. Existe una única cadena que no tiene símbolos denominada **cadena vacía** y denotada por la letra griega ε.[2]

Generalmente se usan las letras u, v y w para denotar cadenas; por ejemplo, la cadena `abbccd` sobre el alfabeto $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ se puede representar por $u = \mathtt{abbccd}$. Usualmente se etiqueta las cadenas finitas comenzando desde la posición inicial 0, por ejemplo la cadena u mencionada anteriormente puede ser denotada por $u = a_0a_1a_2a_3a_4a_5$, donde $a_0 = \mathtt{a}$, $a_1 = \mathtt{b}$, $a_2 = \mathtt{b}$, $a_3 = \mathtt{c}$, $a_4 = \mathtt{c}$ y $a_5 = \mathtt{d}$. Dado que las cadenas se definen como sucesiones (conjuntos sucesionalmente ordenados), el orden de los símbolos en estas es relevante; esto implica, en particular que las cadenas $v = \mathtt{abbc}$ y $w = \mathtt{abcb}$ sean distintas

[1] En este documento $\mathbb{N}$ denota el conjunto de números naturales incluyendo el cero, es decir $\{0, 1, 2, \ldots\}$.

[2] En algunos de los textos de referencia, la cadena vacía es representada por la letra griega λ.

aunque contengan los mismos símbolos.

Ejemplo 1.1.5. Sea $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ el alfabeto que contiene los símbolos `a`, `b` y `c`; entonces, `aab`, `acba`, `acbab` y `ccbaaaaa` son cadenas sobre Σ.

Notación 1.1.6. Dado un alfabeto Σ, se denota por Σ^* al conjunto que contiene todas las cadenas finitas sobre el alfabeto Σ incluyendo a ε y por Σ^+ al conjunto que contiene todas las cadenas no vacías sobre el alfabeto Σ.

Ejemplo 1.1.7. Dado el alfabeto Σ_2, se tiene que:

$$\Sigma_2{}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010 \ldots\},$$

y

$$\Sigma_2{}^+ = \{0, 1, 00, 01, 10, 11, 000, 001, 010 \ldots\}.$$

Definición 1.1.8 (Sub-Cadena, prefijo y sufijo de una cadena)**.** Sean Σ un alfabeto y $u \in \Sigma^*$, una cadena sobre Σ, entonces:

(1) Una cadena $w \in \Sigma^*$ es **sub-cadena** o **sub-palabra** de u si existen cadenas $x, y \in \Sigma^*$ tales que $u = xwy$.

(2) Una cadena $w \in \Sigma^*$ es un **prefijo** de u si existe $v \in \Sigma^*$ tal que $u = wv$; además, w es un **prefijo propio** si $v \neq \varepsilon$.

(3) Una cadena $w \in \Sigma^*$ es un **sufijo** de u si existe $v \in \Sigma^*$ tal que $u = vw$; además, w es un **sufijo propio** si $v \neq \varepsilon$.

La cadena vacía ε es sub-cadena, prefijo y sufijo de cualquier cadena, y a su vez toda cadena es sub-cadena, prefijo y sufijo de sí misma.

Ejemplo 1.1.9. Sea $\Sigma = \{\texttt{a},\texttt{b},\texttt{c}\}$ un alfabeto y la cadena $u = \texttt{abcac}$ en Σ^*, entonces se tiene que:

Prefijos de u:	**Sufijos de** u:	**Sub-cadenas de** u:
ε	ε	Prefijos de u
a	c	Sufijos de u
ab	ac	b
abc	cac	bc,ca
abca	bcac	bca
abcac	abcac	

Notación 1.1.10. Sea Σ un alfabeto, $u = a_0a_1\cdots a_i\cdots a_j\cdots a_n \in \Sigma^*$ y dos números $i,\ j \in \mathbb{N}$ con $0 \leq i \leq n$ y $i-1 \leq j \leq n$, entonces se denota por $u[i]$ al símbolo en la posición i en la cadena u, en este caso, $u[i] = a_i$, y se denota por $u[i..j]$ a la sub-cadena de u que empieza por el símbolo en la posición i y contiene todos los siguientes símbolos hasta la posición j, es decir, $u[i..j] = a_ia_{i+1}\cdots a_j$. Como j puede tomar el valor de i e $i-1$ en la definición anterior, para estos casos se tiene que; $u[i..i] = u[i] = u_i$ y que $u[i..i-1] = \varepsilon$.

Definición 1.1.11 (Longitud de cadenas)**.** Sea Σ un alfabeto y $u \in \Sigma^*$. La **longitud** de u, denotada por $|u|$, es la cantidad de símbolos que conforman a u; es decir:

$$|u| = \begin{cases} 0, & \textbf{si } u = \varepsilon; \\ n, & \textbf{si } u = a_0a_1\cdots a_{n-1}, \textbf{ con } a_i \neq \varepsilon \textbf{ para } i = 0,1,\ldots,n-1. \end{cases}$$

Ejemplo 1.1.12. La cadena $u = 1000110$ sobre el alfabeto Σ_2 tiene longitud 7; esto es $|u| = 7$.

Ejemplo 1.1.13. Sea $\Sigma = \{\texttt{a},\texttt{b},\texttt{c}\}$ un alfabeto, entonces las cadenas en Σ^* que tienen a lo más longitud 3 son:

- **Cadenas de longitud** 0**:** ε.
- **Cadenas de longitud** 1**:** a, b, c.
- **Cadenas de longitud** 2**:** aa, ab, ac, ba, bb, bc, ca, cb y cc.
- **Cadenas de longitud** 3**:** aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb y ccc.

Notación 1.1.14. Sean Σ un alfabeto, $u \in \Sigma^*$ y a un símbolo de Σ, se denota por $|u|_a$ a la cantidad de veces que aparece el símbolo a en la concatenación de símbolos de la cadena u.

Ejemplo 1.1.15. Si $u = \texttt{1010010111}$, entonces $|u| = 10$, $|u|_0 = 4$ y $|u|_1 = 6$.

Definición 1.1.16 (Concatenación de cadenas)**.** Sea Σ un alfabeto y dos cadenas $u, v \in \Sigma^*$. La **concatenación** de u y v denotada por $u \cdot v$ (o simplemente uv), se define recursivamente como:

(i) **Base:** Si $u = \varepsilon$, entonces $uv = \varepsilon v = v\varepsilon = v$.

(ii) **Paso recursivo:** Sea v una cadena con $|v| = n \geq 1$. Entonces $v = wa$, para alguna cadena $w \in \Sigma^*$ con $|w| = n - 1$ y un símbolo $a \in \Sigma$, y se tiene que $uv = (uw)a$.

En otras palabras, se puede reescribir la definición anterior como sigue: Si $u, v \in \Sigma^*$ entonces,

(i) Si $u = \varepsilon$, entonces $uv = \varepsilon v = v\varepsilon = v$.

(ii) Si $u = a_0 a_1 \cdots a_n$ y $v = b_0 b_1 \cdots b_m$, entonces $uv = a_0 a_1 \cdots a_n b_0 b_1 \cdots b_m$.

Ejemplo 1.1.17. Sea $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ un alfabeto y $u = \texttt{abcd}$, $v = \texttt{ebca}$ dos cadenas en Σ^*, entonces $uv = \texttt{abcdebca}$ y $vu = \texttt{ebcaabcd}$.

Proposición 1.1.18 (Propiedades de la concatenación de cadenas)**.** Sea Σ un alfabeto, entonces la concatenación de cadenas sobre Σ satisface las siguientes propiedades:

(a) **Cerradura:** $uv,\ vu \in \Sigma^*$, para todo $u, v \in \Sigma^*$.

(b) **Asociativa:** $(uv)w = u(vw)$, para todo $u, v, w \in \Sigma^*$.

(c) **Existencia del elemento neutro:** Existe $\varepsilon \in \Sigma^*$ tal que $u\varepsilon = \varepsilon u = u$.

(d) $|uv| = |vu| = |u| + |v|$, para todo $u, v \in \Sigma^*$.

De las partes (a), (b) y (c) de la Proposición 1.1.18, se tiene que el conjunto Σ^* con la operación concatenación tiene estructura de **monoide**. Por otro lado, si $u, v \in \Sigma^*$, se tiene en general que $uv \neq vu$; como se vio en Ejemplo 1.1.17.

Definición 1.1.19 (Reflexión de cadenas)**.** Sea Σ un alfabeto y $u \in \Sigma^*$. La **reflexión** o **inversa** de u, denotada por u^R, se define de la siguiente manera:

$$u^R = \begin{cases} \varepsilon, & \textbf{si } u = \varepsilon; \\ a_n \cdots a_1 a_0, & \textbf{si } u = a_0 a_1 \cdots a_n. \end{cases}$$

Trivialmente, $|u| = |u^R|$.

Ejemplo 1.1.20. Sea $u = \texttt{abcdabab}$ una cadena sobre el alfabeto $\Sigma = \{\texttt{a},\texttt{b},\texttt{c}\}$, entonces $u^R = \texttt{babadcba}$.

Ejemplo 1.1.21. Sea $u = \texttt{abccba}$ una cadena sobre el alfabeto $\Sigma = \{\texttt{a},\texttt{b},\texttt{c}\}$, entonces $u^R = \texttt{abccba}$. En este ejemplo se puede observar que $u = u^R$, las cadenas que cumplen esta propiedad se denominan **palíndromos**.

Proposición 1.1.22 (Propiedades de la reflexión de cadenas)**.** Sea Σ un alfabeto, entonces la reflexión de cadenas sobre un alfabeto Σ satisface las siguientes propiedades:

(a) $(u^R)^R = u$, para todo $u \in \Sigma^*$.

(b) $(uv)^R = v^R u^R$, para todo $u, v \in \Sigma^*$.

Definición 1.1.23 (Potencia n-ésima de una cadena)**.** Sean Σ un alfabeto, $u \in \Sigma^*$, una cadena sobre Σ, y $n \geq 0$ un número entero. La **potencia n-ésima** de u, denotada por u^n, se define recursivamente de la siguiente manera:

(i) **Base:** $u^0 = \varepsilon$.

(ii) **Paso recursivo:** $u^n = (u^{n-1})u$, con $n \geq 1$.

Ejemplo 1.1.24. Sea $u = \texttt{ab}$ una cadena sobre el alfabeto $\Sigma = \{\texttt{a},\texttt{b}\}$, entonces $u^3 = \texttt{ababab}$.

Proposición 1.1.25 (Propiedades de la operación potencia n-ésima de una cadena)**.** Sea Σ un alfabeto, entonces la potenciación de cadenas sobre Σ cumple las siguientes propiedades:

(a) $u^0 = \varepsilon$, para todo $u \in \Sigma^*$

(b) $u^{i+j} = u^i u^j$, para todo $u \in \Sigma^*$.

(c) $|u^k| = k|u|$, para todo $u \in \Sigma^*$ y $k \in \mathbb{N}$.

1.2. Cadenas infinitas

En esta sección vamos a considerar a las **cadenas infinitas**, las cuales a diferencia de las cadenas finitas mostradas en la Definición 1.1.4, se definen como una **sucesión infinita** (o secuencia infinita) de símbolos elegidos sobre un alfabeto Σ.

Definición 1.2.1 (Cadenas infinitas)**.** Sea Σ un alfabeto y u una cadena sobre Σ. Se dice que u es una cadena **infinita a derecha** sobre Σ, si puede ser descrita como una sucesión infinita $(u_n)_{n\geq 0}$ de términos o valores en $\Sigma \setminus \{\varepsilon\}$ y con dominio $\mathbb{N}$. Análogamente, se define una cadena **infinita a izquierda** como una sucesión infinita con términos del $\Sigma \setminus \{\varepsilon\}$ y con dominio en el conjunto $\mathbb{Z}^-$. En este documento la mención de cadenas infinitas hará referencia a las cadenas infinitas a derecha, a menos que se especifique otra cosa. Las cadenas infinitas también pueden verse como una concatenación infinita de cadenas finitas sobre el alfabeto $\Sigma \setminus \{\varepsilon\}$; es decir, una cadena u es infinita a derecha sobre Σ si:

$$u = \prod_{i\geq 0} w_i,$$

con $w_i \in \Sigma^* \setminus \{\varepsilon\}$.

Además, si $w_0, w_1, w_2, \ldots$ es una secuencia de cadenas, donde w_i es prefijo propio de w_j para todo $0 \leq i < j$, entonces

$$u = \lim_{n\to\infty} w_n,$$

es la única palabra para la cual las palabras $w_0, w_1, w_2, \ldots$ son prefijos.

Ejemplo 1.2.2 (Cadena infinita)**.** Sea $\Sigma = \{a, b\}$ un alfabeto, entonces la cadena

$$u = (u_n)_{n\geq 0} = abbaabbaabbaabba\cdots$$

es una cadena infinita sobre el alfabeto Σ.

Notación 1.2.3. Sea Σ un alfabeto. Se denota por Σ^w al conjunto de todas las cadenas infinitas sobre Σ.

Definición 1.2.4 (Función característica)**.** Sea $S \subseteq \mathbb{N}$ un conjunto, la **función característica** de S, denotada por f_S, es la función $f_S : \mathbb{N} \longrightarrow \Sigma_2$ definida como:

$$f_S(n) = \begin{cases} 1, & \textbf{si } n \in S; \\ 0, & \textbf{si } n \notin S. \end{cases}$$

Ejemplo 1.2.5 (Sucesión característica de cuadrados perfectos)**.**

$$q = (q_n)_{n\geq 0} = 110010000010000001\cdots$$

Note que si S es el conjunto de cuadrados perfectos en $\mathbb{N}$, entonces $q_n = f_S(n)$, para todo $n \in \mathbb{N}$.

Observación 1.2.6. En algunas ocasiones al definir una cadena infinita es necesario sustituir el subíndice de inicio. La cadena del ejemplo anterior se definía a partir del subíndice 0 pero en ocasiones este se puede cambiar como en el siguiente ejemplo.

Ejemplo 1.2.7 (Sucesión característica de los números primos)**.**

$$p = (p_n)_{n\geq 1} = 0110101000101\cdots$$

Si S es en conjunto fijo de números primos, entonces la función característica $f_S(n)$ relacionada con p_n es:

$$f_s(n) = \begin{cases} 1, & \text{si } n \text{ es primo;} \\ 0, & \text{de lo contrario.} \end{cases}$$

Ejemplo 1.2.8 (Sucesión característica de los números enteros múltiplos de 2)**.**

$$u = (u_n)_{n\geq 0} = 1010101010101010\cdots$$

Si $S = \{n \in \mathbb{N} : \ n = 2k, \text{ para todo } k \in \mathbb{N}\}$, entonces la función característica relacionada con u_n es:

$$f_s(n) = \begin{cases} 1, & \textbf{si } n \equiv 0 \mod 2; \\ 0, & \textbf{si } n \equiv 1 \mod 2. \end{cases}$$

Definición 1.2.9 (Prefijo de una cadena infinita)**.** Sean Σ un alfabeto y $u = (a_n)_{n\geq 0}$ una cadena infinita sobre Σ. Una cadena v sobre Σ es **prefijo** de u si existe otra cadena infinita $w = (b_n)_{n\geq 0}$ sobre Σ tal que $u = v \cdot w$.

Notación 1.2.10. Sea Σ un alfabeto. Se denota por Σ^∞ al conjunto de todas las cadenas finitas e infinitas sobre Σ, es decir, $\Sigma^\infty = \Sigma^* \cup \Sigma^w$.

Definición 1.2.11 (Frecuencia de un símbolo en una palabra)**.** Sea Σ un alfabeto, un símbolo $a \in \Sigma$ y una cadena infinita $u \in \Sigma^w$, entonces si el límite

$$\lim_{n\to\infty} \frac{|u[0..n-1]|_a}{n}$$

existe y es igual a r, se dice que la frecuencia del símbolo a en la cadena infinita u es r y se denota como $Freq_a(u)$. Es decir; $Freq_a(u) = r$.

Ejemplo 1.2.12. Sea u la cadena infinita del Ejemplo 1.2.8, entonces se puede ver que $Freq_1(u) = Freq_0(u) = \frac{1}{2}$.

Definición 1.2.13 (Cadenas estrictamente periódicas)**.** Sea Σ un alfabeto y una cadena finita v sobre Σ. La cadena $u = vvvv\cdots$ que resulta de concatenar v infinitas veces es denominada **estrictamente periódica** y la cadena v es denominado **periodo** de u. Otra notación para la cadena u con periodo v es $u = v^w$.

Definición 1.2.14 (Cadenas finalmente periódicas)**.** Sea Σ un alfabeto y u una cadena infinita sobre Σ. La cadena u es denominada **finalmente periódica** si es de la forma wv^w con las cadenas $v, w \in \Sigma^+$. Si una cadena u es finalmente periódica sobre Σ, con $u = wv^w$, entonces

(1) La cadena w es denominada **pre-periodo** de u, además, si la longitud del pre-periodo w es la menor posible, entonces w es denominada el **menor pre-periodo** de u.

(2) La cadena v es denominada **periodo** de u, además, si la longitud de v es la menor posible, entonces la cadena v es denominada el **menor periodo** de u.

Ejemplo 1.2.15. Sea $\Sigma = \{0, 1, 2, 4, 5, 7, 8, \text{“.”}\}$ un alfabeto y $u = 0.148271482714\ldots$ una cadena finalmente periódica sobre Σ. La cadena u es la representación decimal de la fracción $\frac{1}{7}$ con menor pre-periodo $w = 0.$ y menor periodo $v = 14827$.

1.3. Lenguajes y expresiones regulares

Definición 1.3.1 (Lenguaje)**.** Sea Σ un alfabeto. Un conjunto L es denominado **lenguaje** sobre Σ si es un sub-conjunto de Σ^*, es decir, $L \subseteq \Sigma^*$.

Todo lenguaje L satisface $\emptyset \subseteq L \subseteq \Sigma^*$ y puede ser finito o infinito. Los lenguajes se denotan por las letras mayúsculas tales como $A, B, \ldots, L, M, \ldots$; Además, $L_\varepsilon = \{\varepsilon\}$ es el lenguaje sobre todos los alfabetos y que únicamente contiene la cadena vacía.

Ejemplo 1.3.2. Sea $\Sigma = \{a, b, c\}$ un alfabeto, entonces el conjunto que contiene las cadenas en Σ^* que comienzan con la letra c, es decir, el conjunto $L = \{ca, cb, cc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc, caaa, \ldots\}$ es un lenguaje sobre Σ.

Ejemplo 1.3.3. Sea $\Sigma = \{a, b, c\}$ un alfabeto, entonces el conjunto $L = \{a^n b : n \in \mathbb{N}\}$ es el lenguaje sobre Σ que sólo contiene contiene las cadenas que inician con alguna potencia de a seguida de la letra b.

Dado que un lenguaje sobre un alfabeto Σ es un subconjunto de Σ^* se tienen las operaciones usuales entre conjuntos y sus propiedades en el contexto de lenguajes. Además, se pueden definir otras operaciones como la que se muestra en el literal (5) de la Definición 1.3.4.

Definición 1.3.4 (Operaciones entre lenguajes)**.** Sea Σ un alfabeto y L_1, L_2 dos lenguajes sobre Σ.

(1) La **unión** entre L_1 y L_2, denotada por $L_1 \cup L_2$, es el lenguaje dado por:

$$L_1 \cup L_2 = \{u \in \Sigma^* : u \in L_1 \vee\ u \in L_2\}.$$

(2) La **intersección** entre L_1 y L_2, denotada por $L_1 \cap L_2$, es el lenguaje dado por:

$$L_1 \cap L_2 = \{u \in \Sigma^* : u \in L_1 \wedge\ u \in L_2\}.$$

(3) La **diferencia** entre L_1 y L_2, denotada por $L_1 \setminus L_2$, es el lenguaje dado por:

$$L_1 \setminus L_2 = \{u \in \Sigma^* : u \in L_1 \wedge\ u \notin L_2\}.$$

(4) El **complemento** de L_1, denotado por L_1^c, es el lenguaje dado por:

$$L_1^c = \Sigma^* \setminus L_1 = \{u \in \Sigma^* : u \notin L_1\}.$$

(5) La **concatenación** (o producto) de L_1 y L_2, denotado por L_1L_2, es el lenguaje dado por:

$$L_1L_2 = \{w = vu \in \Sigma^* : v \in L_1 \wedge u \in L_2\}.$$

Ejemplo 1.3.5. Sea $\Sigma = \{a,b,c\}$ un alfabeto y $L_1 = \{aa,bb,cc,abc,bac,cac\}$ y $L_2 = \{a,b,ab,aa,ac,ca,cb,cc,bac\}$ dos alfabetos sobre Σ, entonces se tiene que:

(a)
$$L_1 \cup L_2 = \{a,b,aa,ab,ac,bb,ca,cb,cc,abc,bac,cac\}.$$

(b)
$$L_1 \cap L_2 = \{aa,cc,bac\}.$$

(c)
$$L_1 \setminus L_2 = \{bb,abc,cac\}.$$

(d)
$$L_1L_2 = \{aaa,aab,\ldots,caccc,cacbac\}.$$

A continuación se muestran algunas propiedades para la operación concatenación.

Proposición 1.3.6 (Propiedades de la concatenación de lenguajes)**.** Sea Σ un alfabeto. La concatenación de lenguajes sobre Σ satisface las siguientes propiedades:

(a) $L\emptyset = \emptyset L = \emptyset$, para todo $L \subseteq \Sigma^*$.

(b) **Cerradura:** L_1L_2 es un lenguaje sobre Σ, para todo $L_1,\ L_2 \subseteq \Sigma^*$.

(c) **Asociativa:** $L_1(L_2L_3) = (L_1L_2)L_3$, para todo $L_1,\ L_2,\ L_3 \subseteq \Sigma^*$.

(d) **Existencia del elemento neutro:** El lenguaje $L_\varepsilon = \{\varepsilon\}$ es denominado elemento neutro para la concatenación de lenguajes, ya que $L_\varepsilon L = LL_\varepsilon = L$, para todo lenguaje $L \subseteq \Sigma^*$.

(e) **Distributiva de la concatenación respecto a la unión:** Para todo L_1, L_2, $L_3 \subseteq \Sigma^*$, se cumple que:

$$L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3 \quad \text{y} \quad L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3.$$

(e) **Distributiva de la concatenación respecto a la intersección:** Para todo L_1, L_2, $L_3 \subseteq \Sigma^*$, se cumple que:

$$L_1(L_2 \cap L_3) = L_1L_2 \cap L_1L_3 \quad \text{y} \quad L_1(L_2 \cap L_3) = L_1L_2 \cap L_1L_3.$$

Ejemplo 1.3.7. Sea $\Sigma = \{a,b,c,d,e\}$ y sean $L_1 = \{aba, bb, ab\}$, $L_2 = \{ec, cb, eab\}$ lenguajes sobre Σ, entonces:

$$L_1L_2 = \{abaec, abacb, abaeab, bbec, \ldots, abeab\},$$

y

$$L_2L_1 = \{ecaba, ecbb, ecab, \ldots, eabab\}.$$

Definición 1.3.8 (Potencia n-ésima de un lenguaje)**.** Sea Σ un alfabeto, L un lenguaje sobre Σ y $n \in \mathbb{N}$, entonces la **potencia n-ésima** de L, denotada por L^n, se define recursivamente como:

(i) **Base:** $L^0 = L_\varepsilon = \{\varepsilon\}$

(ii) **Paso recursivo:** $L^n = (L^{n-1})L$, para $n \geq 1$.

Notación 1.3.9 (Cadenas infinitas sobre un lenguaje)**.** Sea Σ un alfabeto y L un lenguaje contenido en Σ^*, se denota por L^w al conjunto

$$L^w = \{w_1w_2w_3\cdots : \ w_i \in L \setminus \{\varepsilon\} \ \text{ para todo } i \geq 0\}.$$

Notación 1.3.10. Sea Σ un alfabeto, L un lenguaje sobre Σ y $n \geq 1$ un entero, entonces se denota por $L^{\leq n}$, el conjunto formado por la unión de todas las potencias de las cadenas en L hasta la potencia n-ésima, es decir

$$L^{\leq n} = L^0 \cup L^1 \cup \cdots \cup L^n.$$

Definición 1.3.11 (Otras operaciones entre lenguajes)**.** Sea Σ un alfabeto y L un lenguaje sobre Σ.

(1) La **clausura de Kleene** de L, denotada por L^*, se define como

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

el conjunto de todas las concatenaciones de cadenas en L.

(2) La **clausura positiva** de L, denotada por L^+, se define como

$$L^+ = LL^* = \bigcup_{i=1}^{\infty} L^i.$$

(3) El **lenguaje reverso o la reflexión de un lenguaje** L, denotado por L^R, se define como:

$$L^R = \{u \in \Sigma^* : u^R \in L\}.$$

Ejemplo 1.3.12. Sea $\Sigma = \{a,b,c\}$ un alfabeto y $L = \{aa, ab, ac\}$ un lenguaje sobre Σ, entonces se tiene que:

(a)
$$L^* = \{\varepsilon, aa, ab, ac, aaaa, aaab, aaac, \ldots\}.$$

(b)
$$L^+ = \{aa, ab, ac, aaaa, aaab, aaac, \ldots\}.$$

(c)
$$L^R = \{aa, ba, ca\}.$$

Definición 1.3.13 (Cociente entre lenguajes)**.** Sea Σ un alfabeto y L_1, L_2 dos lenguajes sobre Σ, el **cociente** entre L_1 y L_2, denotado por L_1/L_2, se define como:

$$L_1/L_2 = \{u \in \Sigma^* : \exists v \in L_2 \text{ tal que } uv \in L_1\}. \tag{1.1}$$

Ejemplo 1.3.14. Sea $L_1 = \{abcacc, abbaa, acab, abc, acb\}$ y $L_2 = \{a, aa, c, cc\}$, entonces el cociente $L_1/L_2 = \{ab, abb, abba, abca, abcac\}$.

Todos los lenguajes que se pueden formar usando las operaciones unión, concatenación y estrella de Kleene, a partir de los lenguajes básicos : $\emptyset, \{\varepsilon\}, \{a\}$ para todo $a \in \Sigma$, con Σ un alfabeto, se denominan **lenguajes regulares**. Por ejemplo:

Ejemplo 1.3.15. $L = \{0,1\}^*\{1\}$ es el lenguaje regular sobre Σ_2 el cual agrupa a todas las cadenas sobre el alfabeto Σ_2 que terminan en el símbolo 1.

Definición 1.3.16 (Lenguajes regulares)**.** Sea Σ un alfabeto. Un **lenguaje regular** sobre Σ se define recursivamente de la siguiente manera:

(i) **Base:** Los lenguajes $\emptyset$, $\{\varepsilon\}$ y $\{a\}$ para cada símbolo $a \in \Sigma$, son lenguajes regulares sobre Σ; los cuales también son denominados lenguajes regulares básicos.

(ii) **Paso recursivo:** Sean L_1 y L_2 lenguajes regulares sobre Σ, entonces los lenguajes $L_1 \cup L_2$, L_1L_2 y L_1^* también son lenguajes regulares sobre Σ.

Se dice entonces que un lenguaje L sobre un alfabeto Σ es regular si y solo si se puede obtener a partir de un número finito de aplicaciones del paso recursivo. Las operaciones unión, la concatenación y la estrella de Kleen, sobre lenguajes que se definieron en la sección anterior y se utilizan en el paso recursivo anterior se denominan **operaciones regulares**.

Una **expresión regular** sobre un alfabeto Σ es una cadena finita y bien formada sobre el alfabeto:

$$\Sigma^\bigstar = \Sigma \cup \{\varepsilon, \emptyset, (,), \cup, ^*\}$$

que se utiliza para representar lenguajes regulares. Por ejemplo, si $\Sigma = \{a,b\}$ es un alfabeto, entonces la palabra sobre el alfabeto $\Sigma^\bigstar$

$$a \cup (a \cup b)^2,$$

es la expresión regular que representa el lenguaje regular

$$\{a, aa, ab, ba, bb\}.$$

Definición 1.3.17 (Expresiones regulares sobre un alfabeto)**.** Sea Σ un alfabeto. Una expresión regular sobre Σ se define recursivamente como:

(i) **Base:** El símbolo $\emptyset$ representa la expresión regular para el lenguaje $\emptyset$, el símbolo ε representa la expresión regular para el lenguaje $\{\varepsilon\}$ y el símbolo a es una expresión regular que representa al lenguaje $\{a\}$ para todo $a \in \Sigma$.

(ii) **Paso recursivo:** Si a y b son dos expresiones regulares sobre el alfabeto Σ, entonces:

(1) La **concatenación** de las expresiones regulares a y b, denotada por (ab), se define como:

$$(ab) = \{a\}\{b\} = \{ab\},$$

también es una expresión regular.

(2) La **unión** de las expresiones regulares a y b, denotada por $(a \cup b)$, se define como:

$$(a \cup b) = \{a\} \cup \{b\} = \{a, b\},$$

también es una expresión regular.

(3) La **clausura de Kleene** de la expresión regular a, denotada por $(a)^*$, se define como:

$$(a)^* = \bigcup_{i=0}^{\infty} \{a\}^i = \{\varepsilon, a, a^2, \ldots\}$$

.

Luego cualquier cadena definida sobre $\Sigma^\bigstar$ que pueda ser generada por un paso finito de iteraciones de la inducción de la base y el paso recursivo anterior es una expresión regular sobre el alfabeto Σ.

Notación 1.3.18 (Lenguaje generado por una expresión regular)**.** Sea Σ un alfabeto y u una expresión regular sobre Σ, entonces $\mathscr{L}(u)$ denota el lenguaje generado (o especificado) por la expresión regular u.

Proposición 1.3.19 (Expresiones y lenguajes regulares)**.** Sea Σ un alfabeto. Un lenguaje L sobre Σ es regular, si existe una expresión regular sobre Σ, $u \in (\Sigma^\bigstar)^*$, tal que $L = \mathscr{L}(u)$.

Teorema 1.3.20. *Todo lenguaje finito es regular.*

Demostración: Sea $L = \{w_0, w_1, \ldots, w_n\}$ *un lenguaje finito, entonces* $L = \mathscr{L}(u)$*, donde u es la expresión regular*

$$w_0 \cup w_1 \cup \cdots \cup w_n.$$

■

1.4. Homomorfismos uniformes

De manera general, los homomorfismos son funciones o aplicaciones entre estructuras matemáticas que preservan su estructura interna. En este documento los homomorfismos que se usarán están definidos sobre conjuntos de cadenas, y por ende, la estructura interna que se preserva es aquella generada por la operación concatenación entre cadenas; es decir, la estructura de monoide.

La relación que se enuncia en el título de este documento se desarrolla a partir del método de construcción de cadenas infinitas por medio de una clase de homomorfismos denominados uniformes y que a su vez son prolongables sobre un elemento de su dominio, como se mostrará a continuación.

Definición 1.4.1 (Homomorfismo sobre cadenas)**.** Sean Σ y Δ dos alfabetos. Una función $\varphi : \Sigma^* \longrightarrow \Delta^*$ es denominada un **homomorfismo** (o morfismo) sobre cadenas si para cada par de cadenas $u, v \in \Sigma^*$ se cumple que:

$$\varphi(uv) = \varphi(u)\varphi(v).$$

Donde $\varphi(u)\varphi(v)$ es la concatenación de las imágenes de u y v bajo la función φ; además, si una función $\varphi : \Sigma^* \longrightarrow \Delta^*$ es un homomorfismo sobre cadenas, entonces $\varphi(\varepsilon_\Sigma) = \varepsilon_\Delta$, donde ε_Σ y ε_Δ son las palabras vacías de Σ^* y Δ^* respectivamente.

Observación 1.4.2. Sean Σ y Δ dos alfabetos y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas, entonces, φ es completamente determinado (o puede ser extendido) sobre el conjunto Σ^* a partir de las imágenes $\varphi(a)$ definidas para todo símbolo $a \in \Sigma$; es decir, si $u = u_0 u_1 \ldots u_n \in \Sigma^*$,

entonces $\varphi(u) = \varphi(u_0)\varphi(u_1)\cdots\varphi(u_n) \in \Delta^*$.

Definición 1.4.3 (Parámetros de un homomorfismo sobre cadenas)**.** Sean Σ y Δ dos alfabetos y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas, entonces:

(1) El **ancho** de φ es el máx$\{|\varphi(a)| : a \in \Sigma\}$.

(2) La **profundidad** de φ es el cardinal del alfabeto Σ; es decir, $|\Sigma|$.

(3) El **tamaño** de φ se define como:

$$\sum_{a\in\Sigma} |\varphi(a)|.$$

Ejemplo 1.4.4. Sean $\Sigma = \{a,b,c\}$, $\Delta = \{1,2\}$ y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas, definido por:

$$\varphi(a) := 21,\ \varphi(b) := 121 \text{ y } \varphi(c) := 121212.$$

Si $u = aacba$ una palabra sobre Σ, entonces,

$$\varphi(u) = \varphi(a)\varphi(a)\varphi(c)\varphi(b)\varphi(a) = 212112121212121.$$

Además:

(1) El ancho de φ es máx$\{2,3,6\} = 6$.

(2) La profundidad de φ es $|\{a,b,c\}| = 3$.

(3) El tamaño de φ es 11.

Definición 1.4.5 (Homomorfismo sobre cadenas uniforme)**.** Sean Σ y Δ dos alfabetos y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas, entonces φ es denominado k**-uniforme** (o simplemente uniforme) si existe un entero positivo k tal que $|\varphi(a)| = k$ para todo $a \in \Sigma$, de lo contrario se dice que es no uniforme. Un homomorfismo sobre cadenas 1-uniforme es denominado **codificación** (o traducción).

Ejemplo 1.4.6. El homomorfismo $\varphi : {\Sigma_2}^* \longrightarrow {\Sigma_2}^*$ que envía $0 \to 01$ y $1 \to 10$ es un homomorfismo 2-uniforme y también es conocido como el homomorfismo **Thue-Morse** por su relación con la secuencia que lleva el mismo nombre.

Definición 1.4.7 (Imagen inversa de un homomorfismo sobre cadenas)**.** Sean Σ y Δ dos alfabetos, $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas y L un lenguaje sobre Δ, entonces la **imagen inversa** del homomorfismo φ sobre L, denotado con $\varphi^{-1}(L)$, se define como

$$\varphi^{-1}(L) = \{u \in \Sigma^* : \varphi(u) \in L\}.$$

Un mecanismo bien conocido para generar cadenas infinitas es a partir de iteración de un homomorfismo sobre un punto fijo, de ahí el término de **homomorfismo iterado**. La iteración de un homomorfismo está relacionado con la propiedad de prolongación que es definida a continuación.

Definición 1.4.8 (Prolongación de un homomorfismo)**.** Sean Σ un alfabeto, $\varphi : \Sigma^* \longrightarrow \Sigma^*$ un homomorfismo y $a \in \Sigma$ un símbolo. Se dice que φ es **prolongable** en a, si $\varphi(a) = au$, para alguna cadena $u \in \Sigma^*$ y $\varphi^n(u) \neq \varepsilon$ para todo entero $n \geq 0$.[3]

Definición 1.4.9 (Sucesión puramente mórfica)**.** Sean Σ un alfabeto, una sucesión infinita $(a_n)_{n\geq 0}$ que toma valores de Σ es denominada **puramente mórfica**, si existen un homomorfismo $\varphi : \Sigma^* \longrightarrow \Delta^*$, un símbolo $a_0 \in \Sigma$ y una cadena $u \in \Sigma^*$ tal que:

(i) $\varphi(a_0) = a_0 u$;

(ii) Dado un entero $n \geq 1$, entonces $\varphi^n(u) \neq \varepsilon$;

(iii) La sucesión de cadenas $(\varphi^n(a_0))_{n\geq 0}$ converge a la sucesión $(a_n)_{n\geq 0}$ cuando n tiende a infinito.

Inmediatamente de (i) se puede observar que:

$$\begin{aligned} \varphi(a_0) &= a_0 u; \\ \varphi^2(a_0) = \varphi(\varphi(a_0)) &= \varphi(a_0 u) = \varphi(a_0)\varphi(u) = a_0 u \varphi(u); \\ \varphi^3(a_0) = \varphi(\varphi^2(a_0)) &= \varphi(a_0 u \varphi(u)) = a_0 u \varphi(u) \varphi^2(u). \end{aligned}$$

[3]Donde $\varphi^n(u) = \varphi \circ \varphi^{n-1}(u)$ y $\circ$ representa la composición de funciones.

y en general, por inducción sobre n, se tiene que para $n \geq 2$:

$$\varphi^n(a_0) = a_0 u \varphi(u) \varphi^2(u) \cdots \varphi^{n-1}(u).$$

Notación 1.4.10. Sean Σ un alfabeto y un homomorfismo prolongable en $a_0 \in \Sigma$, entonces se denota por $\varphi^w(a_0)$ a la cadena infinita $a_0 u \varphi(u) \varphi^2(u) \varphi^3(u) \cdots$; es decir,

$$\varphi^w(a_0) := a_0 u \varphi(u) \varphi^2(u) \varphi^3(u) \cdots .$$

Note que:

$$\begin{aligned} \varphi(\varphi^w(a_0)) &= \varphi(a_0\ u\ \varphi(u)\varphi^2(u)\cdots); \\ &= \varphi(a_0)\varphi(u)\varphi(\varphi(u))\varphi(\varphi^2(u))\cdots; \\ &= a_0\ u\ \varphi(u)\varphi^2(u)\cdots; \\ &= \varphi^w(a_0). \end{aligned}$$

Luego, se dice que $\varphi^w(a)$ es un punto fijo de φ debido a que $\varphi(\varphi^w(a)) = \varphi^w(a)$.

Definición 1.4.11 (Sucesión mórfica)**.** Sean Σ un alfabeto, una sucesión infinita $(a_n)_{n\geq 0}$ que toma valores de Σ es denominada **mórfica**, si existe un alfabeto Δ y una sucesión infinita $(b_n)_{n\geq 0}$ que toma valores de Δ tal que:

(i) La sucesión $(b_n)_{n\geq 0}$ es puramente mórfica;

(ii) Existe una codificación $\mu : \Delta^* \longrightarrow \Sigma^*$ que envía la sucesión $(b_n)_{n\geq 0}$ a la sucesión $(a_n)_{n\geq 0}$, es decir, la sucesión $(a_n)_{n\geq 0}$ es la imagen punto a punto de $(b_n)_{n\geq 0}$ bajo μ.

Si el homomorfismo que hace a la sucesión $(b_n)_{n\geq 0}$ mórfica es un k-morfismo, entonces la sucesión $(a_n)_{n\geq 0}$ es denominada k-automática.[4]

Observación 1.4.12 (★). Una sucesión puramente mórfica también se denomina **punto fijo iterativo** del homomorfismo que la genera (por su construcción). Una sucesión mórfica es la

[4]En el tercer capítulo de este texto se abordará más a fondo la caracterización de una sucesión como k-automática, siendo este el tema principal de estudio en este documento.

imagen punto a punto de la codificación de un punto fijo iterativo del homomorfismo que la genera y finalmente una sucesión k-automática es la imagen punto a punto de la codificación de un punto fijo iterativo del homomorfismo uniforme que la genera.

1.5. Representación de números naturales en base k

Sea $k \geq 2$ un entero. La representación usual en base k utiliza los dígitos del conjunto $\Sigma_k = \{0, 1, \ldots, k-1\}$ para representar los números naturales. El siguiente teorema nos garantiza la existencia y unicidad de dicha representación y su prueba nos indica cómo obtenerla.

Teorema 1.5.1. *Cada número natural n tiene representación canónica única*

$$n = \sum_{i=0}^{m} a_i k^i = a_m k^m + a_{m-1} k^{m-1} + \cdots + a_1 k + a_0,$$

donde $m \geq 0$, $a_i \in \Sigma_k$ *para* $0 \leq i \leq m$ *y* $a_m \neq 0$.

Demostración: Por el algoritmo de la división, existen enteros q_1 y a_0 tales que

$$n = q_1 k + a_0,$$

con $0 \leq a_0 \leq k-1$. Si $q_1 \geq k$ se puede dividir una vez más, para obtener $q_1 = q_2 k + a_1$ y entonces

$$n = (q_2 k + a_1)k + a_0 = q_2 k^2 + a_1 k + a_0,$$

con $0 \leq a_0, a_1 \leq k-1$. Si $q_2 < k$, se detiene el proceso y ya se tiene la representación buscada. Si $q_2 \geq k$, se puede continuar en la forma anterior; es decir, en el siguiente paso se divide q_2 por k para obtener $q_2 = q_3 k + a_2$, donde $0 \leq a_2 \leq k-1$, y ahí que

$$n = q_3 k^3 + a_2 k^2 + a_1 k + a_0.$$

Si se continua este proceso se tiene una sucesión estrictamente decreciente de enteros positivos q_i, tales que $n > q_1 > q_2 > q_1 > \cdots \geq 0$. Entonces este proceso debe terminar en algún momento.

Supóngase que se termina en la $(m-1)$-ésima etapa, donde

$$q_{m-1} = q_m k + a_{m-1},$$

con $0 \leq a_{m-1} \leq k-1$ y $0 \leq q_m \leq k-1$. Sea $a_m = q_m$, entonces se tiene que

$$n = a_m k^m + a_{m-1} k^{m-1} + \cdots + a_1 k + a_0.$$

Para demostrar que esta representación es única, supóngase que n tiene dos representaciones diferentes; es decir

$$n = a_m k^m + a_{m-1} k^{m-1} + \cdots + a_1 k + a_0 = c_m k^m + c_{m-1} k^{m-1} + \cdots + c_1 k + c_0,$$

con $0 \leq a_i \leq k-1$ para cada i y $0 \leq c_j \leq k-1$ para cada j (se puede utilizar el mismo subíndice m en las dos representaciones de n simplemente adicionando $a_i = 0$ ó $c_j = 0$ si es necesario). Luego

$$0 = d_m k^m + d_{m-1} k^{m-1} + \cdots + d_2 k^2 + d_1 k + d_0,$$

donde $d_i = a_i - c_i$, para $i = 1, \ldots, m$. Como se supone que las dos representaciones de n son diferentes, se debe tener que $d_i \neq 0$ para algún i. Sea t el menor subíndice tal que $d_t \neq 0$. Entonces

$$0 = d_m k^m + d_{m-1} k^{m-1} + \cdots + d_{k+1} k^{t+1} + d_t k^t,$$

dividiendo por k^t, $d_t = -k(d_m k^{m-t-1} + \cdots + d_{t+1})$. Así $k | d_t$. Pero como $0 \leq a_t < k$ y $0 \leq c_t < k$ se tiene que $|d_t| < k$. entonces la única posibilidad para que $|d_t| < k$ y $k | d_t$ es que $d_t = 0$, lo que es un contradicción. Por lo tanto la representación de n es única. ■

Corolario 1.5.2*. (Representación en base k) *Por consecuencia del Teorema 1.5.1, se tiene que; si $k \geq 2$ es un entero, entonces para todo $n \in \mathbb{N}$, existe una cadena $u = a_0 a_1 \ldots a_m$ en $(\Sigma_k)^*$, tal que*

$$n = a_0 k^m + a_1 k^{m-1} + \cdots + a_{m-2} k^2 + a_{m-1} k + a_m.$$

*Se dice entonces que u es **representación o notación en base** k **de** n y se denota por:*

$$(n)_k = u = a_0 a_1 \dots a_{m-1} a_m.$$

Además, si $u = a_0 a_1 \dots a_m$ *es una cadena en* ${\Sigma_k}^*$ *entonces se denota por* $[u]_k$ *a la combinación lineal:*

$$[u]_k = a_0 k^{n-1} + \dots + a_{m-1} k + a_m.$$

Finalmente de lo anterior, se tiene que $[(m)_k]_k = m$.

De lo anterior se puede concluir también, que todo entero no negativo tiene una única representación para una cadena en el lenguaje regular C_k definido como:

$$C_k := \{\varepsilon\} \cup (\Sigma_k \setminus \{0\}){\Sigma_k}^*.$$

Observación 1.5.3. La convención general, si no se dice lo contrario, para las representaciones numéricas en base k es que son escritas de izquierda a derecha comenzando con el dígito diferente de cero que multiplica a la mayor potencia de k y se descartan las representaciones con ceros a la izquierda.

Ejemplo 1.5.4. La cadena $u = 10000 \in {\Sigma_2}^*$ es la representación numérica en base-2 para el número natural 16, es decir $[10000]_2 = 16$ o $(16)_2 = 10000$, esto porque:

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2 + 0 = 16.$$

2. AUTÓMATAS FINITOS

Los autómatas son uno de los modelos computacionales más elementales que existen, y su funcionamiento se equipara a una máquina de estados descrita matemáticamente capaz de reconocer una cadena de símbolos elegidos sobre un alfabeto dado, un conjunto de cadenas sobre un alfabeto dado y entre otros, al igual que lo haría una persona al escuchar a otra persona hablar, entender su mensaje y todas las aplicaciones que esto conlleva.

En este capítulo se introduce al lector en la teoría de autómatas propia de la rama matemática de computación con el fin de poder definir a las secuencias automáticas. Existen diferentes clases autómatas dependiendo del tipo de transición y sus condiciones, aquí se presentan algunas de estas clases.

2.1. Autómatas finitos deterministas

Los autómatas finitos deterministas son uno de los modelos de computación más básicos y se pueden describir como un aceptador (o de lo contrario, rechazador) de cadenas o sucesiones finitas sobre un alfabeto dado.

Definición 2.1.1 (Autómata finito determinista)**.** Un **autómata finito determinista** $\mathbb{M}$, o por sus siglas **AFD**, es una maquina de estados finitos que se define como una quíntupla

$$\mathbb{M} = (Q, \Sigma, \delta, q_0, F), \tag{2.1}$$

donde,

(1) $Q = \{q_0, q_1, \ldots, q_n\}$, es un conjunto finito, cuyos elementos son denominados **estados**.

(2) Σ es el alfabeto sobre el cual las cadenas de entrada están definidas.

(3) $\delta : Q \times \Sigma \longrightarrow Q$ es una función, denominada **función de transición**.

(4) $q_0 \in Q$ es el **estado inicial**.

(5) $\emptyset \neq F \subseteq Q$ es el conjunto de **estados finales** o de aceptación.

Note que la función de transición δ puede extenderse al dominio $Q \times \Sigma^*$ de la siguiente forma: Para todo $q \in Q$ y $v \in \Sigma^*$,

$$\delta(q,v) = \begin{cases} q, & \text{si } v = \varepsilon; \\ \delta(q,a), & \text{si } v = a, \text{ con } a \in \Sigma; \\ \delta(\delta(q,u),a), & \text{si } v = ua, \text{ con } u \in \Sigma^* \text{ y } a \in \Sigma. \end{cases}$$

Sea $u = a_0 a_1 \cdots a_n$ una cadena sobre Σ y $\mathbb{M}$ un AFD como en (2.1). El proceso de reconocimiento de u a través de $\mathbb{M}$ inician leyendo de izquierda a derecha los símbolos de la cadena de entrada $a_0 a_1 \cdots a_n$, empezando por a_0 y a partir de la función de transición δ y el estado inicial q_0, es decir, $\delta(q_0, a_0)$ decide cuál es el siguiente estado a evaluar con el siguiente símbolo a_1, es decir $\delta(q_0, a_0 a_1)$ y así sucesivamente hasta finalizar la lectura y transición de todos los símbolos de la cadena de entrada u en el autómata $\mathbb{M}$; si el estado alcanzado $\delta(\delta(q_0, a_0 a_1 \cdots a_{n-1}), a_n)$ está en el conjunto de estados finales F, entonces se dice que la cadena u es aceptada por el autómata $\mathbb{M}$, de lo contrario se dice que la cadena u es rechazada, por esta razón se denomina autómata finito determinista, debido a que si dada una cadena sobre el alfabeto de entrada Σ del autómata $\mathbb{M}$, el autómata determina si la cadena es aceptada o rechazada.

Ejemplo 2.1.2. Sea el AFD $\mathbb{M} = \{Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b, c\}, \delta, q_0, F = \{q_2\}\}$, donde la función $\delta : Q \times \Sigma \longrightarrow Q$ está definida y tabulada en la siguiente tabla:

δ	a	b	c
q_0	q_0	q_1	q_0
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_0

Tabla 2.1

Tabulación de la función de transición δ en el Ejemplo 2.1.2.

Se probará que la cadena $u = acbbc$ es aceptada por $\mathbb{M}$, para ello:

$$\begin{aligned}\delta(q_0,u) &= \delta(q_0,acbbc);\\ &= \delta(\delta(q_0,a),cbbc);\\ &= \delta(\delta(q_0,c),bbc);\\ &= \delta(\delta(q_0,b),bc);\\ &= \delta(\delta(q_1,b),c);\\ &= \delta(q_1,c);\\ &= q_2 \in F.\end{aligned}$$

Luego la cadena u es aceptada por $\mathbb{M}$, sin embargo, la cadena $v = acbbca$ no es aceptada, esto es porque:

$$\begin{aligned}\delta(q_0,v) &= \delta(q_0,ua);\\ &= \delta(\delta(q_0,u),a);\\ &= \delta(\delta(q_0,acbbc),a);\\ &= \delta(q_2,a);\\ &= q_0 \notin F.\end{aligned}$$

Definición 2.1.3 (Lenguaje aceptado por un AFD dado)**.** Sea $\mathbb{M} = (Q,\Sigma,\delta,q_0,F)$ un AFD, el lenguaje aceptado por $\mathbb{M}$ es el conjunto de todas las cadenas aceptadas o reconocidas por $\mathbb{M}$, y se denota como $\mathscr{L}(\mathbb{M})$. Formalmente se define como

$$\mathscr{L}(\mathbb{M}) = \{u \in \Sigma^* : \delta(q_0,u) \in F\}.$$

Ejemplo 2.1.4. El lenguaje aceptado por el AFD $\mathbb{M}$ del Ejemplo 2.1.2 es:

$$\mathscr{L}(\mathbb{M}) = \{u \in (\{a,b,c\})^*, \text{ tal que } u \text{ termina con el sufijo } bc\}.$$

Un AFD puede ser representado por medio de un grafo dirigido denominado **diagrama de transición**. Los diagramas de transición denotan los estados $Q = \{q_0,q_1,\ldots,q_n\}$ por medio de circunferencias y con aristas dirigidas, que salen de ellas hacia otras circunferencias o estados y que acepta bucles de acuerdo a la función de transición. El estado inicial q_0 es representado y diferenciado de los demás con una flecha marcada con la palabra inicio y los estados finales

son dibujados con una doble circunferencia.

Ejemplo 2.1.5. El AFD $\mathbb{M} = (Q = \{q_0, q_1\}, \Sigma_2 = \{0,1\}, \delta, q_0, F = \{q_0\})$ cuyo lenguaje aceptado $\mathscr{L}(\mathbb{M})$ contiene las cadenas u sobre el alfabeto Σ_2 que tienen un número par de 1s en su concatenación, esto es, $\mathscr{L}(\mathbb{M}) = \{u \in {\Sigma_2}^* : |u|_1 = 2k \text{ con } k \in \mathbb{N}\}$, en la Tabla 2.2 se muestra la tabulación de la función de transición δ y en la Figura 2.1 el diagrama de transición para $\mathbb{M}$.

Entonces se puede decir que, si dada una cadena $u \in \Sigma^*$, si $\delta(q_0, u) \in F$, entonces u tiene un número par de unos en su concatenación de símbolos, y en caso contrario, si $\delta(q_0, u) \in Q \setminus F$, entonces u tiene un número impar de unos en su concatenación. Por ejemplo, si $u = 1001 \in {\Sigma_2}^*$ entonces

$$\begin{aligned}\delta(q_0, u) = \delta(q_0, 1001) &= \delta(\delta(q_0, 1), 001);\\ &= \delta(q_1, 001) = \delta(\delta(q_1, 0), 01);\\ &= \delta(q_1, 01) = \delta(\delta(q_1, 0), 1);\\ &= \delta(q_1, 1) = q_0 \in F.\end{aligned}$$

Y así $u \in \mathscr{L}(\mathbb{M})$, sin embargo si $v = u1 = 10011 \in \Sigma^*$ y se tiene que

$$\begin{aligned}\delta(q_0, v) &= \delta(\delta(q_0, u), 1);\\ &= \delta(q_0, 1) = q_1 \in Q \setminus F.\end{aligned}$$

Así, la cadena v con un número impar de unos en su concatenación de símbolos.

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_0

Tabla 2.2

Tabulación de la función de transición δ en el Ejemplo 2.1.5.

Definición 2.1.6 (Estado alcanzable)**.** Sea $\mathbb{M} = (Q, \Sigma, \delta, q_0, F)$ un AFD, entonces se dice que

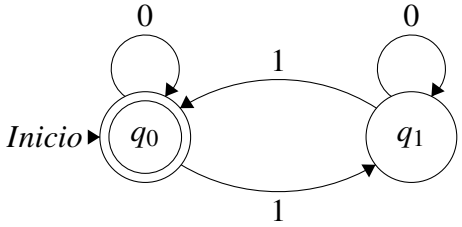


Figura 2.1

Diagrama de transición del AFD $\mathbb{M}$ en el Ejemplo 2.1.5.

un estado $q \in Q$ es **alcanzable** si existe $u \in \Sigma^*$ tal que $\delta(q_0,u) = q$, de lo contrario, se denomina inalcanzable.

Teorema 2.1.7. *Sea $L = \mathscr{L}(\mathbb{M})$ el lenguaje aceptado por el por el AFD $\mathbb{M} = (Q,\Sigma,\delta,q_0,F)$ con k estados, entonces el lenguaje $\overline{L} = \Sigma^* \setminus L$ es el lenguaje aceptado por un AFD con k estados.*
Demostración: Considerando $\mathbb{M}' = (Q,\Sigma,\delta,q_0,Q \setminus F)$, con Q,Σ,δ,q_0 y F iguales a los dados en la hipótesis para $\mathbb{M}$, entonces se probará que $\mathscr{L}(\mathbb{M}') = \overline{L}$. En efecto, para toda cadena $u \in \overline{L}$ se tiene que:

$$\begin{aligned}
u \in \Sigma^* \setminus L &\iff u \in \Sigma^* \text{ y } u \notin L;\\
&\iff u \text{ no es aceptado por } \mathbb{M};\\
&\iff \delta(q_0,u) \in Q \text{ y } \delta(q_0,u) \notin F;\\
&\iff \delta(q_0,u) \in Q \setminus F;\\
&\iff u \text{ es aceptado por } \mathbb{M}';\\
&\iff u \in \mathscr{L}(\mathbb{M}').
\end{aligned}$$

De esta forma se prueba la doble contenencia, es decir, $\mathscr{L}(\mathbb{M}') = \overline{L} = \Sigma^* \setminus L$ y $\mathbb{M}'$ tiene k estados.[1] ■

Teorema 2.1.8. *Sea $L_1 = \mathscr{L}(\mathbb{M}_1)$ y $L_2 = \mathscr{L}(\mathbb{M}_2)$ los lenguajes aceptados por los AFD $\mathbb{M}_1 = (Q_1,\Sigma,\delta_1,q_{0_1},F_1)$ y $\mathbb{M}_2 = (Q_2,\Sigma,\delta_2,q_{0_2},F_2)$, con m y n estados respectivamente, entonces $L_1 \cap L_2$ y $L_1 \cup L_2$ son los lenguajes aceptados por dos AFD con $m \times n$ estados.*
Demostración: Considerando $\mathbb{M}' = (Q = Q_1 \times Q_2,\ \Sigma,\ \delta,\ q_0 = (q_{0_1},\ q_{0_2}),\ F = F_1 \times F_2)$[2] *y $\mathbb{M}'' = (Q = Q_1 \times Q_2,\ \Sigma,\delta,q_0 = (q_{0_1},q_{0_2}),\ F' = ((F_1 \times Q_2) \cup (Q_1 \times F_2)))$ dos AFD con*

[1] Esta demostración fue estudiada de [3, pág 129] y complementada en este documento.
[2] $Q_1 \times Q_2$ denota el producto directo entre Q_1 y Q_2.

función de transición δ definida para todo $(q_1,q_2) \in Q_1 \times Q_2$ y $u \in \Sigma^*$ como:

$$\delta((q_1,q_2),u) = (\delta_1(q_1,u),\delta_2(q_2,u)).$$

Se probará que $L_1 \cap L_2 = \mathscr{L}(\mathbb{M}')$, esto es, para toda cadena $u \in L_1 \cap L_2$ se tiene que:

$$\begin{aligned} u \in L_1 \cap L_2 &\iff u \in L_1 \text{ y } u \in L_2; \\ &\iff u \in \mathscr{L}(\mathbb{M}_1) \text{ y } u \in \mathscr{L}(\mathbb{M}_2); \\ &\iff \delta_1(q_{0_1},u) \in F_1 \text{ y } \delta_2(q_{0_2},u) \in F_2; \\ &\iff (\delta_1(q_{0_1},u),\delta_2(q_{0_2},u)) \in F_1 \times F_2; \\ &\iff \delta((q_{0_1},q_{0_2}),u) \in F; \\ &\iff u \in \mathscr{L}(\mathbb{M}'). \end{aligned}$$

De esta forma $\mathscr{L}(\mathbb{M}') = L_1 \cap L_2$. Ahora se probará que $L_1 \cup L_2 = \mathscr{L}(\mathbb{M}'')$, esto es, que para toda cadena $u \in L_1 \cup L_2$ se tiene que:

$$\begin{aligned} u \in L_1 \cup L_2 &\iff u \in L_1 \text{ o } u \in L_2; \\ &\iff u \in \mathscr{L}(\mathbb{M}_1) \text{ o } u \in \mathscr{L}(\mathbb{M}_2); \\ &\iff \delta_1(q_{0_1},u) \in F_1 \text{ o } \delta_2(q_{0_2},u) \in F_2; \\ &\iff \delta((q_{0_1},q_{0_2}),u) \in F_1 \times Q_1 \text{ o } \delta((q_{0_1},q_{0_2},u)) \in Q_1 \times F_2; \\ &\iff \delta((q_{0_1},q_{0_2}),u) \in ((F_1 \times Q_2) \cup (Q_1 \times F_2)); \\ &\iff \delta((q_{0_1},q_{0_2}),u) \in F'; \\ &\iff u \in \mathscr{L}(\mathbb{M}''). \end{aligned}$$

de ahí que $\mathscr{L}(\mathbb{M}'') = L_1 \cup L_2$.[3] ■

Ejemplo 2.1.9. Sea $\mathbb{M}_1 = (Q_1 = \{p_0,p_1\},\Sigma_2,\delta_1,p_0,F_1 = \{p_0\})$ y $\mathbb{M}_2 = (Q_2 = \{q_0,q_1\},\Sigma_2,\delta_2,q_0,F_2 = \{q_0\})$ dos AFD, donde las funciones de transición $\delta_1 : Q_1 \times \Sigma_2 \longrightarrow Q_1$ y $\delta_2 : Q_2 \times \Sigma_2 \longrightarrow Q_2$ están definidas como:

δ_1 evaluada en $(p_i,a) \in Q_1 \times \Sigma_2$ es:

$$\delta_1(p_i,a) = \begin{cases} p_0, & \textbf{si } (p_i,a) \in \{(p_0,1),(p_1,1)\}; \\ p_1, & \textbf{si } (p_i,a) \in \{(p_0,0),(p_1,0)\}; \end{cases}$$

[3] Esta demostración fue estudiada de [3, pág 129 - 130].

y δ_2 evaluada en $(q_i, a) \in Q_2 \times \Sigma_2$ es:

$$\delta_2(q_i,a) = \begin{cases} q_0, & \textbf{si } (q_i,a) \in \{(q_0,0),(q_1,1)\}; \\ q_1, & \textbf{si } (q_i,a) \in \{(q_0,1),(q_1,0)\}. \end{cases}$$

En la Figura 2.2 se muestra el diagrama de transición para el autómata $\mathbb{M}_1$ y anteriormente se mostró en la Figura 2.1 el diagrama de transición para el autómata $\mathbb{M}_2$ donde $\mathscr{L}(\mathbb{M}_1) = L_1$ son las cadenas sobre Σ_2 que terminan en 1 y $\mathscr{L}(\mathbb{M}_2) = L_2$ son las cadenas sobre Σ_2 con un número par de 1s en su concatenación.

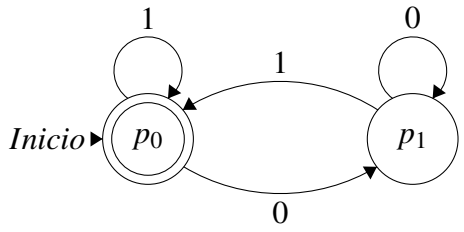


Figura 2.2

Diagrama de transición AFD $\mathbb{M}_1$ en el Ejemplo 2.1.9.

Los AFD $\mathbb{M} = (Q, \Sigma_2, \delta, (p_0,q_0), F)$ y $\mathbb{M}' = (Q, \Sigma_2, \delta, (p_0,q_0), F')$, donde $Q = Q_1 \times Q_2 = \{(p_0,q_0),(p_0,q_1),(p_1,q_0),(p_1,q_1)\}$, $\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a)) \in Q$, $F = \{(p_0,q_0)\}$ y $F' = \{(p_0,q_0),\ (p_0,q_1),\ (p_1,q_0)\}$ cumplen que $\mathscr{L}(\mathbb{M}) = L_1 \cap L_2$ y $\mathscr{L}(\mathbb{M}') = L_1 \cup L_2$. La Tabla 2.3 muestra la tabulación de la función δ para ambos autómatas.

El diagrama de transición para el AFD $\mathbb{M}$ se muestra en la Figura 2.3, cuyo lenguaje aceptado $\mathscr{L}(\mathbb{M})$ es el conjunto que contiene las cadenas sobre Σ_2, que tienen un número par de 1s en su concatenación y terminan en 1 simultáneamente. Además, bastaría con dibujar los estados $[p_0,q_1]$ y $[p_1,q_0]$ como estados aceptadores en la Figura 2.3 para hacerlo coincidir con el diagrama de transición del AFD $\mathbb{M}'$. La Tabla 2.3 muestra la tabulación de la función δ para ambos autómatas.

δ	0	1
$(p_0, q_0]$	$(p_1, q_0]$	(p_0, q_1)
$(p_0, q_1]$	$(p_1, q_1]$	(p_0, q_0)
$(p_1, q_0]$	$(p_1, q_0]$	(p_0, q_1)
$(p_1, q_1]$	$(p_1, q_1]$	(p_0, q_0)

Tabla 2.3

Tabulación de la función de transición δ en el Ejemplo 2.1.9.

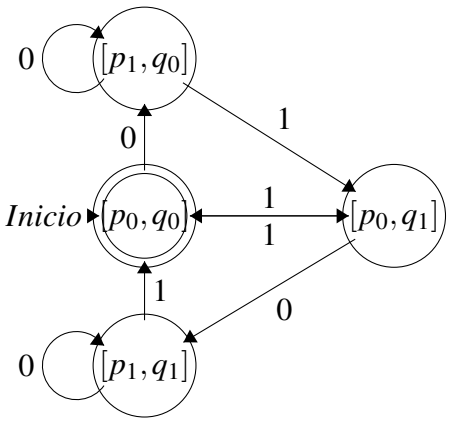


Figura 2.3

Diagrama de transición AFD $\mathbb{M}$ en el Ejemplo 2.1.9.

2.2. Autómatas finitos no deterministas

Ahora, en esta sección se introduce un caso más general de autómatas finitos que son útiles gracias a que permiten tener una representación concisa de ciertos lenguajes.

Definición 2.2.1 (Autómata finito no determinista)**.** Un **autómata finito no determinista** o por sus siglas **AFND** es definido como una quíntupla

$$\mathbb{M} = (Q, \Sigma, \gamma, q_0, F), \tag{2.2}$$

donde,

(1) $Q = \{q_0, q_1, \ldots, q_n\}$, es un conjunto finito, cuyos elementos son denominados **estados**.

(2) Σ es el alfabeto sobre el cual las cadenas de entrada están definidas.

(3) $\gamma : Q \times \Sigma \longrightarrow \mathscr{P}(Q)$ una función [4], denominada **función de transición**.

(4) $q_0 \in Q$ es el **estado inicial**.

(5) $F \neq \emptyset \subseteq Q$ es el conjunto de **estados finales** o de aceptación.

La función de transición $\gamma : Q \times \Sigma \longrightarrow \mathscr{P}(Q)$ es definida para todo par $(q_i, a) \in Q \times \Sigma$ como

$$\gamma(q,a) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}, \text{ con } q_{i_j} \in Q, \text{ para } 0 \leq j \leq k.$$

La función γ se extiende inicialmente a conjuntos de estados, es decir, si $a \in \Sigma$ y $S \subseteq Q$, entonces $\gamma(S,a)$ se define como:

$$\gamma(S,a) = \bigcup_{q \in S} \gamma(q,a).$$

Finalmente la función de transición γ puede extenderse al dominio $\mathscr{P}(Q) \times \Sigma^*$, para todo $S \subseteq Q$ ($S \in \mathscr{P}(Q)$ y $v \in \Sigma^*$, γ se define como:

$$\gamma(S,v) = \begin{cases} S, & \text{si } v = \varepsilon; \\ \bigcup_{q \in S} \gamma(q,a), & \text{si } v = a, \text{ con } a \in \Sigma; \\ \gamma(\gamma(S,u),a) = \bigcup_{q \in \gamma(S,u)} \gamma(q,a), & \text{si } v = ua, \text{ con } u \in \Sigma^* \text{ y } a \in \Sigma. \end{cases}$$

Los autómatas finitos no determinísticos o AFND son similares a los AFD, sin embargo, en este caso los AFND permiten cero, dos o más opciones de estados a elegir al momento de evaluar un mismo par (estado y símbolo) en su función de transición γ. Aunque la forma de representación por diagramas de transición es similar, ahora varias aristas que salen de un mismo estado, pueden estar etiquetadas con un mismo símbolo y llegar a estados diferentes.

Definición 2.2.2 (Lenguaje aceptado por un AFND dado)**.** Sea $\mathbb{M} = (Q, \Sigma, \gamma, q_0, F)$ un AFND, el lenguaje aceptado por $\mathbb{M}$ es el conjunto de todas las cadenas aceptadas o reconocidas por él

[4] $\mathscr{P}(Q)$ (o también denotado por 2^Q) denota el conjunto partes o potencia de Q, y es el conjunto que contiene todos los posibles sub-conjuntos del conjunto Q.

y se denota como $\mathscr{L}(\mathbb{M})$. Formalmente se tiene que

$$\mathscr{L}(\mathbb{M}) = \{u \in \Sigma^* : \gamma(q_0, u) \cap F \neq \emptyset\}.$$

Notación 2.2.3. Dados un autómata finito $\mathbb{M}$, una cadena u sobre el alfabeto de entrada de $\mathbb{M}$ y ρ la función de transición de $\mathbb{M}$, entonces la imagen de $\rho(q_0, u)$, para q_0 el estado inicial de $\mathbb{M}$, es denotada por $\mathbb{M}(u)$, es decir, $\mathbb{M}(u) = \rho(q_0, u)$.
Si $\mathbb{M}$ es un AFND, entonces $\mathbb{M}(u) \subseteq \mathscr{P}(Q)$, con Q el conjunto de estados de $\mathbb{M}$, lo que significa que $\mathbb{M}$ valida la cadena u y devuelve como imagen todos los posibles estados alcanzables por medio de las transiciones de la validación de u a partir de γ, la función de transición de $\mathbb{M}$.

Teorema 2.2.4. *Si L es el lenguaje aceptado por un AFND $\mathbb{M} = (Q, \Sigma, \gamma, q_0, F)$ con n estados, entonces L es el lenguaje aceptado por un AFD con a lo máximo 2^n estados.*
Demostración: Considerando el AFD $\mathbb{M}' = (Q', \Sigma, \delta, q_0', F')$, donde $Q' \subseteq \mathscr{P}(Q)$ es el conjunto de estados, $q_0' = \{q_0\}$ es el estado inicial, $F' = \{S \in Q' : S \cap F \neq \emptyset\}$ el conjunto de estados finales y $\delta : Q' \times \Sigma \longrightarrow Q'$ es la función de transición definida para todo $(S, a) \in Q' \times \Sigma$ como:

$$\delta(S, a) = \bigcup_{q \in S} \gamma(q, a).$$

Se probará que $\mathscr{L}(\mathbb{M}) = \mathscr{L}(\mathbb{M}')$. En efecto, para toda cadena $u \in \mathscr{L}(\mathbb{M})$ se cumple que:

(i) Si $|u| = 0$; es decir, $u = \varepsilon$, entonces

$$\begin{aligned}
u \in \mathscr{L}(\mathbb{M}) &\iff \mathbb{M}(u) \cap F \neq F; \\
&\iff \gamma(q_0, u) \cap F \neq \emptyset; \\
&\iff \exists q \in (\gamma(q_0, \varepsilon) \cap F), \text{ en particular } q = q_0, \text{ dado que } \gamma(q_0, \varepsilon) = q_0; \\
&\iff (\bigcup_{q \in \{q_0\}} \gamma(q, \varepsilon)) \cap F \neq \emptyset; \\
&\iff \delta(q_0', u) \cap F \neq \emptyset; \\
&\iff \delta(q_0', u) \in F', \text{ porque } \delta(q_0', u) \in Q'; \\
&\iff u \in \mathscr{L}(\mathbb{M}').
\end{aligned}$$

(ii) Si $|u| = n \geq 1$, entonces existen $v \in \Sigma^*$ con $|v| = n - 1$ y $a \in \Sigma$ tal que $u = va$, entonces

$$
\begin{aligned}
u \in \mathscr{L}(\mathbb{M}) &\iff \gamma(q_0, va) \cap F \neq \emptyset; \\
&\iff \gamma(\gamma(q_0, v), a) \cap F =\neq \emptyset; \\
&\iff \exists q \in \gamma(q_0, v) \text{ tal que } \gamma(q, a) \cap F \neq \emptyset; \\
&\iff (\bigcup_{q \in \gamma(q_0, v)} \gamma(q, a)) \cap F \neq \emptyset; \\
&\iff \delta(\gamma(q_0, v), a) \cap F \neq \emptyset; \\
&\iff \delta(\delta(q_0{}', v), a) \cap F \neq \emptyset; \\
&\iff \delta(q_0{}', va) \cap F \neq \emptyset; \\
&\iff \delta(q_0{}', u) \in F; \\
&\iff u \in \mathscr{L}(\mathbb{M}').
\end{aligned}
$$

Por lo tanto se tiene que $\mathscr{L}(\mathbb{M}) = \mathscr{L}(\mathbb{M}')$*, con* $\mathbb{M}$ *un AFND con n estados y* $\mathbb{M}'$ *un AFD con* $|Q'| \leq |\mathscr{P}(Q)|$ *estados, es decir, como* $|\mathscr{P}(Q)| = 2^{|Q|} = 2^n$*, entonces* $|Q'| \leq 2^n$ *estados.* [5] ■

Ejemplo 2.2.5. El siguiente AFDN $\mathbb{M}$, representado por el diagrama de transición mostrado en la Figura 2.4, acepta el lenguaje $L = a^*b^*c^*$.

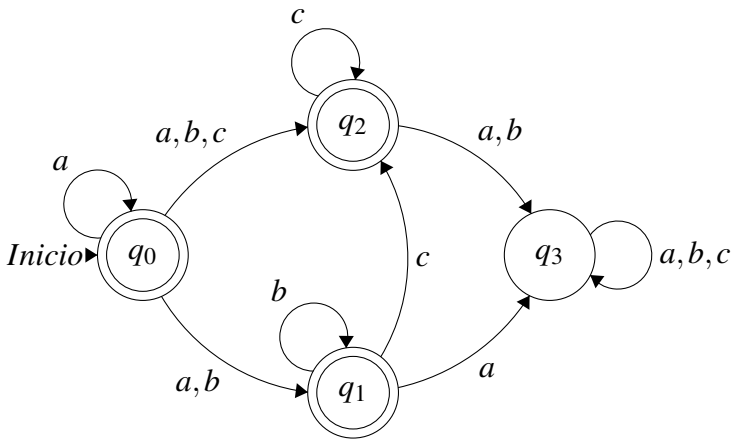


Figura 2.4

Diagrama de transición AFND $\mathbb{M}$ *en el Ejemplo 2.2.5.*

A partir del AFND $\mathbb{M} = (Q, \Sigma, \gamma, q_0, F)$, donde $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$, $F = \{q_0, q_1, q_2\}$ y la función de transición $\gamma : Q \times \Sigma \longrightarrow \mathscr{P}(Q)$ está tabulada en la Tabla 2.4 y el Teorema 2.2.4, se puede construir un AFD $\mathbb{M}'$ tal que $\mathscr{L}(\mathbb{M}') = L$.

[5] Esta demostración fue estudiada de [3, pág 130] y complementada en este documento.

γ	a	b	c
q_0	$\{q_0, q_1\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{q_3\}$	$\{q_1\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$

Tabla 2.4

Tabulación de la función de transición γ en el Ejemplo 2.2.5.

Considerando $\mathbb{M}'$ como $(Q',\ \Sigma,\ \delta,\ p_0,\ F')$, donde Σ, el alfabeto de entrada, es igual que en $\mathbb{M}$ y $Q' = \{p_0, p_1, \ldots, p_8\} \subseteq \mathscr{P}(Q)$ es el conjunto de estados, ${q_0}' = \{q_0\}$ es el estado inicial, $F' = \{S \in Q' :\ S \cap F \neq \emptyset\}$ el conjunto de estados finales y la función $\delta : Q' \times \Sigma \longrightarrow Q'$ es la función de transición definida para todo $(S,a) \in Q' \times \Sigma$ como:$\delta(S,a) = \bigcup_{q \in S} \gamma(q,a)$. La función δ es tabulada en :

δ	a	b	c
p_0	p_1	p_2	p_3
p_1	p_4	p_2	p_3
p_2	p_5	p_6	p_3
p_3	p_5	p_5	p_3
p_4	p_4	p_7	p_8
p_5	p_5	p_5	p_5
p_6	p_5	p_6	p_8
p_7	p_5	p_6	p_8
p_8	p_5	p_5	p_8

Tabla 2.5

Tabulación de la función de transición δ en el Ejemplo 2.2.5.

Donde: $p_0 = \{q_0\}$, $p_1 = \{q_0, q_1\}$, $p_2 = \{q_1, q_2\}$, $p_3 = \{q_2\}$, $p_4 = \{q_0, q_1, q_3\}$, $p_5 = \{q_3\}$, $p_6 = \{q_1, q_3\}$, $p_7 = \{q_1, q_2, q_3\}$, $p_8 = \{q_2, q_3\}$ y $\mathbb{M}'$ es representado por el diagrama de transición mostrado en la Figura 2.5.

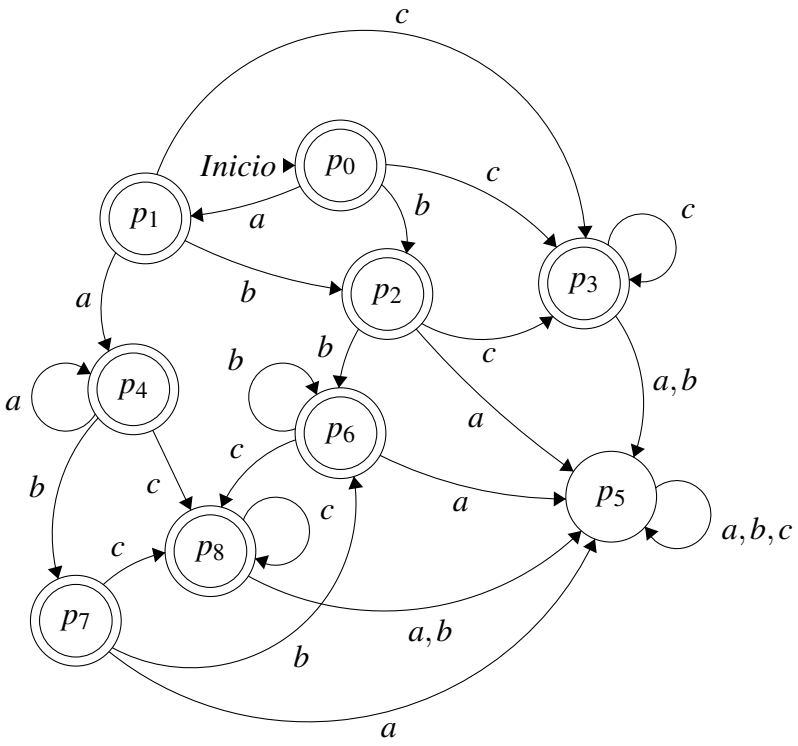


Figura 2.5

Diagrama de transición AFD $\mathbb{M}'$ en el Ejemplo 2.2.5.

2.3. Teorema de Kleene y Lema de Bombeo

En esta sección se presenta el Teorema de Kleene, el cual es una herramienta muy importante a la hora de relacionar los lenguajes regulares y los lenguajes aceptados por un autómata finito, además también se presenta el Lema de Bombeo el cual permite ver cuando esta equivalencia no puede darse.

Para demostrar el siguiente resultado es necesario definir una tercera clase de autómatas finitos denominada autómatas finitos no determinísticos con transiciones λ (particularmente en este documento se tiene que $\lambda = \varepsilon$) o por sus siglas AFND-ε, después de definir esta tercera clase de autómatas finitos es posible demostrar la equivalencia computacional entre los AFD, los AFND y los AFND-ε a partir de los lenguajes que aceptan, los cuales son precisamente los lenguajes regular y a partir de este resultado obtener el teorema conocido como Teorema de Kleene, por esta razón se omite su demostración y se refiere a la cita [3, pág 131 - 137] para su revisión.

Teorema 2.3.1 (Teorema de Kleene)**.** *Un lenguaje L es aceptado por un AFD si y sólo si L es regular.*

Teorema 2.3.2. *Sea Σ un alfabeto. Si L_1 es un lenguaje regular sobre Σ y L_2 es arbitrario sobre Σ, entonces L_1/L_2 es regular.*[6]

Demostración: Como L_1 es un lenguaje regular, entonces por el Teorema de Kleene, existe un AFD $\mathbb{M} = (Q, \Sigma, \delta, q_0, F)$ tal que $\mathscr{L}(\mathbb{M}) = L_1$. Considerando el AFD $\mathbb{M}' = (Q, \Sigma, \delta, q_0, F')$, tal que $F' = \{q \in Q : \exists v \in L_2 \text{ tal que } \delta(q, v) \in F\}$. Se probará que $L_1/L_2 = \mathscr{L}(\mathbb{M}')$, en efecto, puesto que para todo $u \in L_1/L_2$ se tiene que:

$$\begin{aligned} u \in L_1/L_2 &\iff \exists v \in L_2 \text{ tal que } uv \in L_1 = \mathscr{L}(\mathbb{M}); \\ &\iff \exists v \in L_2 \text{ tal que } \delta(q_0, uv) \in F; \\ &\iff \exists v \in L_2 \text{ tal que } \delta(q, v) \in F, \text{ donde } q = \delta(q_0, u) \in Q; \\ &\iff q = \delta(q_0, u) \in F'; \\ &\iff u \in \mathscr{L}(\mathbb{M}') \end{aligned}$$

Así $L_1/L_2 = \mathscr{L}(\mathbb{M}')$ y por el Teorema de Kleene, L_1/L_2 es regular.[7] ■

Teorema 2.3.3. *Sean Σ y Δ dos alfabetos, si L es un lenguaje regular sobre Δ y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo sobre cadenas, entonces $\varphi^{-1}(L)$ es un lenguaje regular sobre Σ.*

Demostración: Como L es un lenguaje regular, entonces por el Teorema de Kleene existe un AFD $\mathbb{M} = (Q, \Delta, \delta, q_0, F)$ tal que $\mathscr{L}(\mathbb{M}) = L$. Considerando el AFD $\mathbb{M}' = (Q, \Sigma, \delta', q_0, F)$, donde $\delta' : Q \times \Sigma \longrightarrow Q$ se define para todo $(q, a) \in Q \times \Sigma$ como:

$$\delta'(q, a) := \delta(q, \varphi(a)).$$

[6] L_1/L_2 denota el cociente entre lenguajes definido en 1.3.13.

[7] Esta demostración fue estudiada de [3, pág 133].

Pruébese que $\varphi^{-1}(L) = \mathscr{L}(\mathbb{M}')$. *En efecto, para* $u \in \varphi^{-1}(L)$ *se cumple que:*

$$\begin{aligned} u \in \varphi^{-1}(L) &\iff \varphi(u) \in L = \mathscr{L}(\mathbb{M}); \\ 2 &\iff \delta(q_0, \varphi(u)) \in F; \\ &\iff \delta'(q_0, u) \in F; \\ &\iff u \in \mathscr{L}(\mathbb{M}'). \end{aligned}$$

Así $\varphi^{-1}(L) = \mathscr{L}(\mathbb{M}')$ *con* $\mathbb{M}'$ *AFD, entonces por el Teorema de Kleene,* $\varphi^{-1}(L)$ *es regular.*[8] ■

Definición 2.3.4. Un AFD $\mathbb{M}$ es **mínimo** si tiene el menor número de estados entre todos los AFD $\mathbb{M}'$ que cumplen que $\mathscr{L}(\mathbb{M}') = \mathscr{L}(\mathbb{M})$.

Ejemplo 2.3.5. En la Figura 2.3 se puede observar el diagrama de transición del AFD $\mathbb{M}$ que acepta las cadenas sobre Σ_2 que tienen un número par de 1s en su concatenación y terminan en 1. Este autómata no es mínimo porque al considerar un cambio en la función de transición δ se puede excluir el estado (p_1, q_1) y se tendría un AFD $\mathbb{M}'$ con menos estados que $\mathbb{M}$ tal que $\mathscr{L}(\mathbb{M}') = \mathscr{L}(\mathbb{M})$ como se muestra en la Figura 2.6.

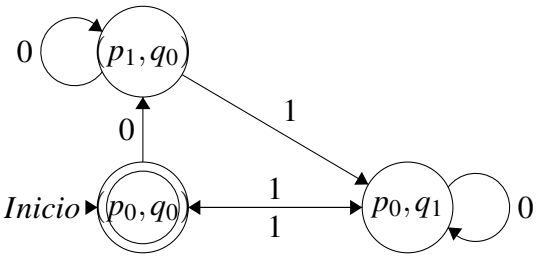


Figura 2.6

Diagrama de transición AFD mínimo $\mathbb{M}$ *en el Ejemplo 2.1.9*

.

Como se mencionó anteriormente el Lema de bombeo que se expone a continuación es una herramienta muy fuerte a la hora de probar que un lenguaje no es regular, además de relacionar también la cantidad mínima de estados necesarios para generar un lenguaje regular.

[8] Esta demostración fue estudiada de [3, pág 133].

Lema 2.3.6*. (Lema de bombeo) *Sea Σ un alfabeto y L un lenguaje regular sobre Σ, entonces existe una constante entera $k \geq 0$, denominada* ***constante de bombeo*** *para L, tal que para toda cadena $z \in L$ con $|z| \geq k$ existe una descomposición $z = uvw$, donde $u, v, w \in \Sigma^*$, $|uv| \leq k$ y $|v| \geq 1$, tal que $uv^i w \in L$ para todo $i \geq 0$. Además, la constante k es el número de estados en el AFD $\mathbb{M}$ mínimo tal que $\mathscr{L}(\mathbb{M}) = L$.*

Demostración: Como L es un lenguaje regular sobre Σ, entonces existe un AFD $\mathbb{M} = (Q, \Sigma, \delta, q_0, F)$ con r estados, para el cual se cumple que $\mathscr{L}(\mathbb{M}) = L$.

Sean $k = r$ y $z = a_0 a_1 \ldots a_{n-1} \in L$, con $|z| = n \geq k$ y $a_i \in \Sigma$ para $0 \leq i \leq n-1$. Considerando la lista de $k+1$ estados $q_0, \delta(q_0, a_0), \delta(q_0, a_0 a_1), \ldots, \delta(q_0, a_0 a_1 \ldots a_{k-1})$, donde δ es la función de transición de $\mathbb{M}$. Entonces, por el principio de palomar, algún estado debe repetirse, es decir aparecer al menos dos veces en la lista. Sea $q_* \in Q$ el primer estado que se repite en la lista, luego existen $0 \leq s < t \leq k-1$ tal que:

$$\delta(q_0, a_0 a_1 \ldots a_s) = q_* = \delta(q_0, a_0 a_1 \ldots a_t), \tag{2.3}$$

entonces haciendo $u = a_0 a_1 \ldots a_s$, $v = a_{s+1} \ldots a_t$ y $w = a_{t+1} \ldots a_{n-1}$ se tiene que $z = uvw$ y $\delta(q_0, u) = \delta(q_0, uv)$, por la igualdad (2.3). Ahora se probará que $uv^i w \in L$ para $i \geq 0$, por inducción sobre i, esto es:

(i) Para $i = 0$, $uv^i w = u\varepsilon w = uw \in L$ porque:

$$\begin{aligned} \delta(q_0, uw) &= \delta(\delta(q_0, u), w); \\ &= \delta(\delta(q_0, uv), w); \\ &= \delta(q_0, uvw) = \delta(q_0, z). \end{aligned}$$

Como $z \in L$, entonces $\delta(q_0, uw) = \delta(q_0, z) \in F$ y $uw \in \mathscr{L}(\mathbb{M}) = L$.

(ii) Suponiendo que $uv^i w \in L$ para $i \geq 1$, hay que probar que $uv^{i+1} w \in L$.

Dado que

$$\begin{aligned} \delta(q_0, uv^{i+1} w) &= \delta(\delta(q_0, uv), v^i w); \\ &= \delta(\delta(q_0, u), v^i w); \\ &= \delta(q_0, uv^i w). \end{aligned}$$

Como $uv^iw \in L$*, entonces* $\delta(q_0, uv^{i+1}w) = \delta(q_0, uv^iw) \in F$ *y así,* $uv^{i+1}w \in \mathscr{L}(\mathbb{M}) = L$.[9] ■

Ejemplo 2.3.7. Sea $\Sigma = \{0,1\}$ un alfabeto. El lenguaje $L = \{1^j 0^{2j} : j \geq 0\}$ sobre Σ no es regular.

Demostración: Por reducción al absurdo, suponiendo que L es regular, entonces existe la constante de bombeo $k \geq 0$ para L, entonces la cadena $z = 1^k 0^{2k} \in L$ se puede descomponer como $z = uvw$, con $|uv| \leq k$ y $|v| \geq 1$. Luego, existen r y s tales que $r+s \leq k$ y $u = 1^r, v = 1^s$ y $w = 1^{k-(r+s)}0^{2k}$. Entonces $uv^0w = 1^{k-s}0^{2k} \notin L$ porque, $|v| = s \geq 1$, lo cual es una contradicción. Por lo tanto, L no es regular. ■

Ejemplo 2.3.8. Sea $\Sigma = \{a\}$ un alfabeto. El lenguaje $L = \{a^{2^n} : n \geq 1\}$ sobre Σ no es regular.

Demostración: Por reducción al absurdo, suponiendo que L es regular, entonces existe la constante de bombeo $k \geq 0$ para L, entonces la cadena $z = a^{2^k} \in L$ se puede descomponer como $z = uvw$, con $|uv| \leq k$ y $|v| \geq 1$, porque $|a^{2^k}| \geq k$. Luego existen r y s tales que $r+s \leq k$ y $u = a^r, v = a^s$ y $w = a^{2^k-(r+s)}$. Entonces $uv^0w = a^r a^{2^k-(r+s)} = a^{2^k-s} \notin L$ porque, $|v| = s \geq 1$, lo cual es una contradicción. Por lo tanto, L no es regular. ■

2.4. Autómatas finitos con salida

Como se vio anteriormente, los AFD sobre un alfabeto Σ, aceptan la cadena $u \in \Sigma^*$ si esta al ser evaluada en la función de transición, a partir del estado inicial, retorna un estado en el conjunto de estados finales. De esta manera se define una partición[10] de Σ^*, determinada por las cadenas que son aceptadas por los autómatas y las que no lo son. De esta forma se puede definir una función característica a partir un AFD que devuelva 1 en caso de que la cadena u sea aceptada por el AFD o 0 en caso contrario. A continuación se define un modelo de autómatas que permite describir de una mejor manera las funciones que puede computar (evaluar o calcular) un AFD.

[9] Esta demostración fue estudiada de [3, pág 136].

[10] Para más detalle Ver Definición 2.4.5

Definición 2.4.1 (Autómata finito determinista con salida)**.** Un **autómata finito determinista con salida** o por sus cifras **AFDS** es definido como una 6-tupla

$$\mathbb{M} = (Q, \Sigma, \delta, q_0, \Delta, \tau), \tag{2.4}$$

donde,

(1) $Q = \{q_0, q_1, \ldots, q_n\}$, es un conjunto finito, cuyos elementos son denominados **estados**.

(2) Σ es el alfabeto sobre el cual las cadenas de entrada están definidas.

(3) $\delta : Q \times \Sigma \longrightarrow Q$ una función, denominada **función de transición**.

(4) $q_0 \in Q$ es el **estado inicial**.

(5) Δ es el alfabeto sobre el cual las cadenas de salida están definidas.

(6) $\tau : Q \longrightarrow \Delta$ una función, denominada **función de salida**.

Si en un AFDS $\mathbb{M}$ se tiene que $\Sigma = \Sigma_k$, con $k \geq 2$, es alfabeto de las cadenas de entrada, entonces se dice que $\mathbb{M}$ es un k-AFDS. Un AFDS $\mathbb{M}$ define o computa una función $f_{\mathbb{M}} : \Sigma^* \longrightarrow \Delta$ de la siguiente forma, para todo $u \in \Sigma^*$

$$f_{\mathbb{M}}(u) = \tau(\delta(q_0, u)). \tag{2.5}$$

Definición 2.4.2. Si una función $f : \Sigma^* \longrightarrow \Delta$ puede ser definida (calculada o computada) de la forma (2.5), para un AFDS $\mathbb{M}$, entonces f se denominada **función de estados finitos**.

Al igual que los AFD, los AFDS pueden ser representados por diagramas de transición con la diferencia en el etiquetado de cada estado, que ahora incluye su imagen de salida bajo la función τ de la forma (q/a), donde $\tau(q) = a$, y además no hay estados finales (aceptadores). Esto se observa en el siguiente ejemplo.

Ejemplo 2.4.3. En el Ejemplo 2.1.5 se muestra el diagrama de transición del AFD $\mathbb{M} = (Q = \{q_0, q_1\}, \Sigma_2, \delta, q_0, F = \{q_0\})$, el cual acepta las cadenas u sobre el alfabeto Σ_2 que tienen un número par de 1s en su concatenación, es decir $\mathscr{L}(\mathbb{M}) = \{u \in \Sigma_2 : |u|_1 = 2k, \text{ con } k \in \mathbb{N}\}$. A partir de $\mathbb{M}$ es posible construir un AFDS

$\mathbb{M}' = \{Q, \Sigma, \delta, q_0, \Delta\}$, donde $Q,\ \Sigma,\ \delta,\ q_0$ son iguales que en $\mathbb{M}$, $\Delta = \{0,1\}$ y $\tau : Q \longrightarrow \Delta$ es la función de salida dada por:

$$\tau(q) = \begin{cases} 0, \text{ si } q = q_0; \\ 1, \text{ si } q = q_1. \end{cases}$$

Entonces, $\mathbb{M}'$ define la función de estados finitos $f_{\mathbb{M}'} : \Sigma^* \longrightarrow \Delta$, dada por:

$$f_{\mathbb{M}'}(u) = \tau(\delta(q_0, u)).$$

El diagrama de transición de $\mathbb{M}'$ es mostrado en la Figura 2.7. Si $u = a_0 \cdots a_{|u|-1} \in {\Sigma_2}^*$, entonces $|u|_1 = \sum_{i=0}^{|u|} a_i = a_0 + \cdots + a_{|u|}$, así entonces se puede enunciar que $f_{\mathbb{M}}(u)$ imprime como salida la suma de los dígitos de una cadena $u \in \Sigma_2$ módulo 2.

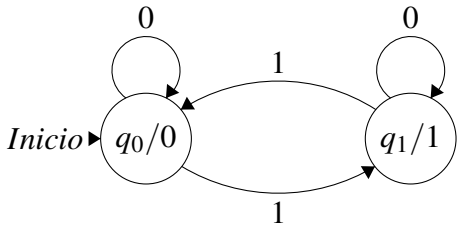


Figura 2.7

Diagrama de transición AFDS $\mathbb{M}'$ en el Ejemplo 2.4.3.

Teorema 2.4.4. *Dado $\mathbb{M} = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ un AFDS, entonces para todo $d \in \Delta$,*

el conjunto

$$I_d(\mathbb{M}) = \{u \in \Sigma^* : \tau(\delta(q_0, u)) = d\} \tag{2.6}$$

es un lenguaje regular.

Demostración: Sea $q \in Q$, considerando el AFD $\mathbb{M}_q = (Q, \Sigma, \delta, q_0, F)$, donde Q, Σ, δ, q_0 son iguales que $\mathbb{M}$, $F = \{q\}$ y $\mathscr{L}(\mathbb{M}_q)$ el lenguaje aceptado por el AFD $\mathbb{M}_q$, entonces se debe probar que:

$$I_d(\mathbb{M}) = \bigcup_{\substack{q \in Q, \\ con\ \tau(q) = d}} \mathscr{L}(\mathbb{M}_q).$$

En efecto, para $u \in I_d(\mathbb{M})$ entonces:

$$\begin{aligned} u \in I_d(\mathbb{M}) &\iff (\exists\, q = \delta(q_0,u) \in Q)\ (\tau(q) = d); \\ &\iff u \in \mathscr{L}(\mathbb{M}_q),\ \textit{donde}\ \mathbb{M}_q = (Q,\Sigma,\delta,q_0,\{q\}); \\ &\iff u \in \bigcup_{\substack{q\in Q,\\ \textit{con}\ \tau(q)=d}} \mathscr{L}(\mathbb{M}_q). \end{aligned}$$

Finalmente, como $\mathscr{L}(\mathbb{M}_q)$ es un lenguaje regular para todo $q \in Q$ y los lenguajes regulares son cerrados bajo la unión finita, entonces $I_d(M)$ es regular.[11] ■

El siguiente teorema utiliza el concepto de partición de un conjunto, el cual se define como:

Definición 2.4.5 (Partición de un conjunto)**.** Dada un conjunto A y $F = \{L_i\}_{i=0}^{k-1}$ una familia de conjuntos, donde $L_i \cap L_j = \emptyset$ para $0 \leq i \neq j < k$. Entonces la familia F es denominada **partición** del conjunto A si se cumple que:

$$\bigcup_{\substack{L_i\in F\\ 0\leq i\leq k-1}} L_i = A.$$

Teorema 2.4.6. *Si $k \geq 1$ es un entero, $\Delta = \{a_0,\ a_1,\ldots,\ a_{k-1}\}$ y Σ dos alfabetos finitos, y $F = \{L_i\}_{i=0}^{k-1}$ una familia de k lenguajes regulares que forman una partición de Σ^*. Entonces existe un AFDS $\mathbb{M} = (Q,\Sigma,\delta,q_0,\Delta,\tau)$ tal que $\tau(\delta(q_0,u)) = a_i \in \Delta$ si y sólo si $u \in L_i$, para $0 \leq i \leq k-1$.*

Demostración: Como L_i es regular para $0 \leq i \leq k-1$, entonces por el teorema de Kleene existe un AFD $\mathbb{M}_i = (Q_i,\Sigma,\delta_i,q_{0i},F_i)$ tal que $\mathscr{L}(\mathbb{M}_i) = L_i$ para $0 \leq i \leq k-1$. Considerando el AFDS $\mathbb{M} = (Q,\Sigma,\delta,q_0,\Delta,\tau)$ donde el conjunto de estados es $Q = Q_1 \times Q_2 \times \cdots \times Q_{k-1}$, el estado inicial $q_0 = (q_{0_0}, q_{0_1}, \ldots, q_{0_{k-1}})$, la función de transición $\delta : Q \times \Sigma \longrightarrow Q$ definida para todo $q = (q_0, q_1, \ldots, q_{k-1}) \in Q$ y $a \in \Sigma$ como:

$$\delta((q_{0_0}, q_{0_1}, \ldots, q_{0_{k-1}}, a) = (\delta_0(q_{0_0}, a), \delta_1(q_{0_1}, a), \ldots, \delta_{k-1}(q_{0_{k-1}}, a)).$$

Finalmente para todo $q = (q_0, q_1, \ldots, q_{k-1}) \in Q$, se define la función de salida $\tau : Q \longrightarrow \Delta$ como: $\tau(q) = a_i$, si existe un único $0 \leq i \leq k-1$ tal que $q_i \in F_i$ y $\tau(q) = a_0$ en otro caso. τ está bien definida puesto que $\{L_i\}_{i=0}^{k-1}$ es una partición de Σ^*. Así entonces para cada $u \in \Sigma^*$, existe un único $0 \leq i \leq k-1$ tal que $u \in L_i$, entonces $u \in \mathscr{L}(\mathbb{M}_i)$, que por definición es $\delta_i(q_{0_i}, u) \in F_i$,

[11] Esta demostración fue estudiada de [3, pág 138] y complementada.

así entonces por la forma en que se define τ *se tiene que* $\tau(\delta_i(q_{0_i}, u)) = a_i$. *Así, entonces se tiene que* $\mathbb{M} = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ *satisface las condiciones del teorema.*[12] ■

Teorema 2.4.7. *Sea* $f : \Sigma^* \longrightarrow \Delta$ *una función de estados finitos, entonces la función* $f^R : \Sigma^* \longrightarrow \Delta$ *definida como* $f^R(u) = f(u^R)$ *también es una función de estados finitos.*

Demostración: Como f *es una función de estados finitos, entonces existe un AFDS* $\mathbb{M} = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ *tal que* $f = f_{\mathbb{M}}$. *Considerando el AFDS* $\mathbb{M}' = \{Q', \Sigma, \delta', q_0', \Delta, \tau'\}$, *donde* $Q' = \Delta^Q$, $q_0' = \tau$, $\tau' : Q' \longrightarrow \Delta$, *tal que para* $g \in Q'$ *se define como:*

$$\tau'(g) := g(q_0).$$

Y la función $\delta' : Q' \times \Sigma \longrightarrow Q'$, *para* $g \in Q'$ *y* $a \in \Sigma$ *se define como:*

$$\delta'(g, a) = h \in Q' \text{ tal que } h(q) = g(\delta(q, a)).$$

Se probará que $\delta'(q_0', u) = h$, *con* $h : Q \longrightarrow \Delta$ *una función definida para todo* $q \in Q$ *como:*

$$h(q) = \tau(\delta(q, u^R)).$$

Por inducción sobre $|u|$,

(i) Si $|u| = 0$, *entonces* $u = \varepsilon$ *y se tendría que:*

$$\begin{aligned} \delta'(q_0', u) = h &\iff h(q) = q_0'(\delta(q, u^R)); \\ &\iff h(q) = \tau(\delta(q, \varepsilon)). \end{aligned}$$

Así, $f_{\mathbb{M}'}(u) = \tau'(\delta'(q_0', u)) = \tau'(h) = h(q_0) = \tau(\delta(q_0, u^R)) = f_{\mathbb{M}}(u^R) = f^R(u)$.

(ii) Supóngase que $\delta'(q_0', u) = h$, *con* $h(q) = \tau(\delta(q, u^R))$ *para todo* $q \in Q$ *y* $|u| = n \geq 1$, *se probará que se cumple para* $|u| = n + 1$, *en efecto, existen* $v \in \Sigma^*$ *y* $a \in \Sigma$ *tal que* $u = va$, *entonces se tiene que:*

[12] Esta demostración fue estudiada de [3, pág 139].

$$\begin{aligned}\delta'(q_0', u) &= \delta'(\delta'(q_0', v), a);\\ &= \delta'(h, a),\ \textit{donde } h(q) = \tau(\delta(q, v^R));\\ &= g,\ \textit{donde } g(q) = h(\delta(q, a)).\end{aligned}$$

Así, se tendría entonces que

$$\begin{aligned}h(\delta(q,a)) &= \tau(\delta(\delta(q,a), v^R));\\ &= \tau(\delta(q, av^R));\\ &= \tau(\delta(q, (va)^R));\\ &= \tau(\delta(q, u^R)).\end{aligned}$$

Como $\tau'(\delta'(q_0', u)) = h(q_0) = \tau(\delta(q_0, u^R))$ *para todo* $u \in \Sigma^*$, *esto es* $f_{M'}(u) = f_M(u^R) = f(u^R) = f^R$ *y así* f^R *es una función de estados finitos.*[13] ■

Corolario 2.4.8*. *Si* $f : \Sigma^* \longrightarrow \Delta$ *es una función de estados finitos computable por un AFDS con n estados, entonces* f^R *es computable por un AFDS con al menos* $|\Delta|^n$ *estados.*

Corolario 2.4.9*. *Si L es un lenguaje regular, entonces* L^R *también es un lenguaje regular.*

Definición 2.4.10 (Traductor de estados finitos)**.** Un **traductor de estados finitos**[14] es una maquina de estados, definida como una 6-tupla

$$\mathbb{T} = (Q, \Sigma, \delta, q_0, \Delta, \lambda) \tag{2.7}$$

donde

(1) $Q = \{q_0, q_1, \ldots, q_n\}$, es un conjunto finito, cuyos elementos son denominados **estados**.

(2) Σ es el alfabeto sobre el cual las cadenas de entrada están definidas.

(3) $\delta : Q \times \Sigma \longrightarrow Q$ una función, denominada **función de transición**.

(4) $q_0 \in Q$ es el **estado inicial**.

(5) Δ es el alfabeto sobre el cual las cadenas de salida están definidas.

(6) $\lambda : Q \times \Sigma \longrightarrow \Delta^*$ una función, denominada **función de salida**.

[13] Esta demostración fue estudiada de [3, pág 139].

[14] Cada vez que en este texto se refiere a un traductor, se asume que es un traductor de estados finitos.

Similar a los AFDS, un traductor $\mathbb{T}$ computa o emula una función $f_{\mathbb{T}} : \Sigma \longrightarrow \Delta^*$, tal que para todo $u \in \Sigma^*$, tal que $u = a_0 a_1 \dots a_k$, se define como:

$$\mathbb{T}(u) = f_{\mathbb{T}}(u) = \prod_{0 \leq i \leq k} \lambda(\delta(q_0, a_0 a_1 \dots a_{i-1}), a_i).$$

Finalmente, un traductor es representado de manera similar a las máquinas que se definieron anteriormente, en este caso no hay estados aceptadores y las transiciones dibujadas de un estado q_i a un estado q_j con grafos ahora serán etiquetadas de la forma a/b, con $a \in \Sigma$ y $b \in \Delta$, que corresponden a la función de transición $\delta(q_i, a) = q_j$ y a la función de salida $\lambda(q_j, a) = b$ respectivamente.

Ejemplo 2.4.11. Sea $\mathbb{T} = (Q, \Sigma_2, \delta, q_0, \Delta, \lambda)$ un traductor donde $Q = \{q_0, q_1\}$, la función δ que coincide con la tabulación presentada en la Tabla 2.6, $\Delta = \{\varepsilon, 0, 1, 2, n\}$ y función λ que coincide con la tabulación presentada en la Tabla 2.7. El traductor $\mathbb{T}$ computa la cantidad de ceros entre cada par de 1s en la concatenación de $u \in \Sigma_2$ cuando es menor que 2. Si la cantidad de ceros entre un par de 1s es mayor que 2 entonces esta imprime n.

δ	0	1
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_0	q_4
q_4	q_0	q_4

Tabla 2.6

Tabulación de la función de transición δ en el Ejemplo 2.4.11.

λ	0	1
q_0	ε	ε
q_1	0	ε
q_2	1	ε
q_3	2	ε
q_4	n	ε

Tabla 2.7

Tabulación de la función de salida λ en el Ejemplo 2.4.11.

Definición 2.4.12 (Traducción y traducción inversa de un lenguaje)**.** Sea $\mathbb{T}$ un traductor, si $L \subseteq \Sigma^*$ y $L' \subseteq \Delta^*$ son lenguajes, entonces

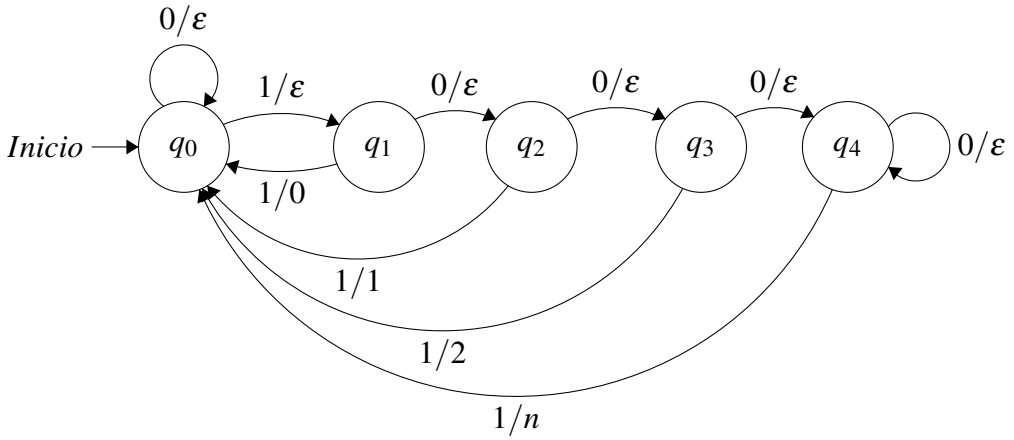


Figura 2.8

Traductor de estados finitos del Ejemplo 2.4.11.

(1) La **traducción** de L, denotado con $\mathbb{T}(L)$ es el conjunto

$$\mathbb{T}(L) := \{f_{\mathbb{T}}(u) : u \in L\}. \tag{2.8}$$

(2) La **traducción inversa** de L', denotado con $\mathbb{T}^{-1}(L')$ es el conjunto

$$\mathbb{T}^{-1}(L') =: \{u \in \Sigma^* : f_{\mathbb{T}}(u) \in L'\}. \tag{2.9}$$

Definición 2.4.13 (Un traductor k-uniforme)**.** Sea $\mathbb{T} = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$ un traductor. $\mathbb{T}$ se denomina **k-uniforme**, si existe un entero $k \geq 1$ tal que $|\lambda(q,a)| = k$, con λ, la función de salida de $\mathbb{T}$, para todo $q \in Q$ y $a \in \Sigma$.

Teorema 2.4.14. *Si L es un lenguaje regular y $\mathbb{T}$ un traductor de estados finitos, entonces $\mathbb{T}(L)$ es un lenguaje regular.*

Demostración: Un bosquejo de esta demostración por construcción es mostrado en [3, pág 141]. ■

Corolario 2.4.15*. *Sea Σ un alfabeto, $L \subseteq \Sigma^*$ un lenguaje regular y $\varphi : \Sigma^* \longrightarrow \Delta^*$ un homomorfismo, entonces $\varphi(L) = \{\varphi(u) : u \in L\}$ es regular.*

Demostración: La demostración es consecuencia directa del Teorema 2.4.14, porque un homomorfismo puede verse como un traductor con exactamente un solo estado. ■

Teorema 2.4.16. *Sean Σ y Δ dos alfabetos, y $\mathbb{T} = (Q',\ \Sigma,\ \delta',\ q_0',\ \Delta,\ \lambda)$ un traductor de estados finitos. Si $L \subseteq \Delta^*$ es un lenguaje regular, entonces $\mathbb{T}^{-1}(L) \subseteq \Sigma^*$ es un lenguaje regular.*

Demostración: Como L es un lenguaje regular, entonces existe un AFD $\mathbb{M} = (Q,\ \Delta,\ \delta,\ q_0,\ F)$ tal que $\mathscr{L}(\mathbb{M}) = L$. Considerando el AFD $\mathbb{M}' = (Q'',\ \Sigma,\ \delta'',\ q_0'',\ F'')$ donde $Q'' = Q' \times Q$, $q_0'' = (q_0', q_0)$, $F'' = Q' \times F$ y la función de transición $\delta'' : Q'' \times \Sigma \longrightarrow Q''$, para todo $(q', q) \in Q''$ y $a \in \Sigma$ se define como:

$$\delta''((q',q),a) = (\delta'(q',a), \delta(q, \lambda(q',a))).$$

Se probará que $\mathbb{T}^{-1}(L) = \mathscr{L}(\mathbb{M}')$. En efecto, para todo $u \in \mathbb{T}^{-1}(L)$, tal que $|u| = k$, es decir $u = a_0 a_1 \ldots a_{k-1}$ se tiene que:

$$\begin{aligned}
u \in \mathbb{T}^{-1}(L) &\iff u \in \Sigma^* \text{ y } f_{\mathbb{T}}(u) = v = \Big(\prod_{0 \le i \le k-1} \lambda(\delta(q_0, a_0 a_1 \ldots a_{i-1}), a_i)\Big) \in L; \\
&\iff v \in \mathscr{L}(\mathbb{M}) \text{ tal que } f_{\mathbb{T}}(u) = v \text{ para } u \in \Sigma^*; \\
&\iff v = \lambda(q_0', a_0)\lambda(\delta(q_0', a_0), a_1) \cdots \lambda(\delta'(q_0', a_0 a_1 \ldots a_{k-2}), a_{k-1}) \in \mathscr{L}(\mathbb{M}), \\
&\quad \text{para } u = a_0 a_1 \ldots a_{k-1} \in \Sigma^*; \\
&\iff v = \lambda(q_0', u) \in \Delta^* \text{ para } u \in \Sigma^*, \text{ tal que } \delta(q_0, v) = \delta(q_0, \lambda(q_0', u)) \in F; \\
&\iff \delta'(q_0', u) \in Q' \text{ y } \delta(q_0, \lambda(q_0', u)) \in F; \\
&\iff (\delta'(q_0', u), \delta(q_0, \lambda(q_0', u))) \in Q' \times F; \\
&\iff \delta''((q_0', q_0), u) = \delta''(q_0'', u) \in Q' \times F = F''; \\
&\iff u \in \mathscr{L}(\mathbb{M}').
\end{aligned}$$

De esta forma, $\mathscr{L}(\mathbb{M}') = \mathbb{T}^{-1}(L)$ y así $\mathbb{T}^{-1}(L)$ es regular.[15] ■

[15] Esta demostración fue estudiada de [3, pág 142].

3. SECUENCIAS AUTOMÁTICAS

En este capítulo se definen las secuencias automáticas, los conjuntos automáticos y se muestra el teorema de Cobham, resultados que dan el título a este documento, para posteriormente en el siguiente capítulo mostrar algunas de sus aplicaciones y relaciones con otras ramas de las matemáticas.

El el capítulo anterior, en la Definición 2.4.2 se definió cuándo una función es denominada función de estados finitos. En este documento se enfoca en aquellas funciones de estados finitos cuyo autómata asociado tiene alfabeto de entrada Σ_k, es decir, un k-AFDS y además las cadenas de entrada representan un número en base k. En base a esto se dice que una secuencia $(a_n)_{n\geq 0}$ es k-automática si cada a_n es la imagen de una función de estados finitos de la representación en base k de n, como se definirá a continuación.

Definición 3.0.1 (Secuencia k-automática)**.** Sea Δ un alfabeto finito, entonces una secuencia $(a_n)_{n\geq 0}$ sobre Δ es denominada **k-automática**, si existe un k-AFDS $\mathbb{M} = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ tal que para todo $a_n \in (a_n)_{n\geq 0}$ con $n \geq 0$, se tiene que $a_n = \tau(\delta(q_0, u))$ para todo $u \in {\Sigma_k}^*$, tal que $[u]_k = n$.[1] Las secuencias k-automáticas también son conocidas, en la literatura de referencia, como k-reconocibles o secuencias de etiquetas uniformes (uniform tag sequences).

Se mostrarán ahora, algunos de los ejemplos característicos o canónicos en esta clase de secuencias y que son bien conocidos en la literatura que se referencia en este documento.

Ejemplo 3.0.2 (Secuencia Thue-Morse)**.** La secuencia

$$u = (a_n)_{n\geq 0} = a_0 a_1 a_2 \cdots = 011010011001 \cdots$$

es llamada la secuencia Thue-Morse y una de sus cualidades radica en las diferentes caracterizaciones que la generan. Una de estas caracterizaciones define que cada a_n toma el valor 1 si la representación binaria del índice n tiene un número impar de unos y el valor 0 si la representación binaria de n tiene un número par de unos, dicho de otra forma, esta secuencia cuenta el número de 1's módulo 2 en la representación de n en base-2.

El AFDS $\mathbb{M}'$ del Ejemplo 2.4.3 en el capítulo anterior es el autómata que genera la secuencia

[1] La notación $[u]_k$ es la combinación linea en base k de u definida en el Corolario 1.5.2.

Thue-Morse, ya que $f_{\mathbb{M}'}((n)_2) = a_n$, para todo $n \geq 0$.

n	0	1	2	3	4	5	6	7	8	9	10
$(n)_2$	0	1	10	11	100	101	110	111	1000	1001	1010
$\|(n)_2\|_1$	0	1	1	2	1	2	2	3	1	2	1
$a_n = \|(n)_2\|_1 \pmod 2$	0	1	1	0	1	0	0	1	1	0	1

Tabla 3.1

Tabulación de los primeros términos de la secuencia Thue-Morse en el Ejemplo 3.0.2.

La secuencia Thue-Morse también es conocida como la secuencia de intercambio más justa y otra de sus caracterizaciones fue mencionada en el Ejemplo 1.4.6, donde puede ser obtenida a partir de un homomorfismo 2-uniforme.

Ejemplo 3.0.3 (Secuencia Golay-Rudin-Shapiro)**.**

$$u = (a_n)_{n\geq 0} = a_0a_1a_2\cdots = 1\ 1\ 1\ -1\ 1\ 1\ -1\ 1\cdots \tag{3.1}$$

Esta secuencia debe su nombre a los estudios independientes de sus propiedades hechos por los matemáticos Marcel Golay, Walter Rudin y Harld Seymour Shapiro. La secuencia Golay-Rudin-Shapiro también es conocida en versión cero-uno por su alfabeto de las cadenas de entrada $\Sigma_2 = \{0,1\}$ y que puede ser descrita como $a_n = |(n)_2|_{11} \pmod 2$.

n	0	1	2	3	4	5	6	7	8	9	10
$(n)_2$	0	1	10	11	100	101	110	111	1000	1001	1010
$\|(n)_2\|_{11}$	0	0	0	1	0	0	1	2	0	0	0
$a_n = \|(n)_2\|_{11} \pmod 2$	0	0	0	1	0	0	1	0	0	0	0

Tabla 3.2

Tabulación de los primeros términos de la secuencia Golay-Rudin-Shapiro en el Ejemplo 3.0.3.

La versión original es definida sobre el alfabeto $\Sigma = \{-1, 1\}$ y en su caracterización se define la función $e_{k;u}(n)$, para todo entero $n \geq 0$ y $u \in {\Sigma_k}^*$, como el número de ocurrencias de la cadena u en la representación canónica de n en la base k (incluyendo superposiciones formadas por superposición[2]), luego la secuencia se puede definir como $a_n = (-1)^{e_{2;11}(n)}$. La secuencia Golay-Rudin-Shapiro es 2-automática ya que es generada por el 2-AFDS representado en el diagrama de transición en Figura 3.1.

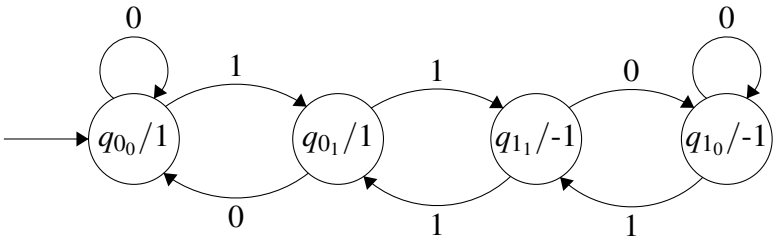


Figura 3.1

Diagrama de transición del AFDS que genera la secuencia Golay-Rudin-Shapiro en el Ejemplo 3.0.3.

Otra característica interesante de la secuencia Golay-Rudin-Shapiro se deriva al dibujar el camino que inicia con un primer movimiento en dirección inicial β y para el paso n, con $n \geq 0$, se decide si ir a la izquierda, girar 90° en dirección contraria a las manecillas del reloj, o derecha, girar 90° en dirección a favor de las manecillas del reloj, de acuerdo a la función b_n que genera una nueva secuencia $(b_n)_{n\geq 0}$ definida a partir de los términos en la secuencia Golay-Rudin-Shapiro $(a_n)_{n\geq 0}$ de la siguiente forma:

$$b_n = \begin{cases} F, & \text{si } n = 0; \\ L, & \text{si } a_n \times a_{n-1} = (-1)^n; \\ R, & \text{si } a_n \times a_{n-1} = -(-1)^n. \end{cases} \tag{3.2}$$

Donde $a_n \times a_{n-1}$ denota la multiplicación habitual en $\mathbb{N}$.

[2]Por ejemplo $(7)_2 = 111$ y $(27)_2 = 11011$ y al contar superposiciones se cumple que $e_{2;11}(7) = e_{2;11}; (27) = 2$.

n	0	1	2	3	4	5	6	7	8	9	10
a_n	1	1	1	-1	1	1	-1	1	1	1	1
b_n	F	R	L	L	R	R	R	L	L	R	L

Tabla 3.3

Tabulación de los primeros diez términos de la secuencia b_n en el Ejemplo 3.0.3, cuya gráfica extendida genera la Figura 3.2 .

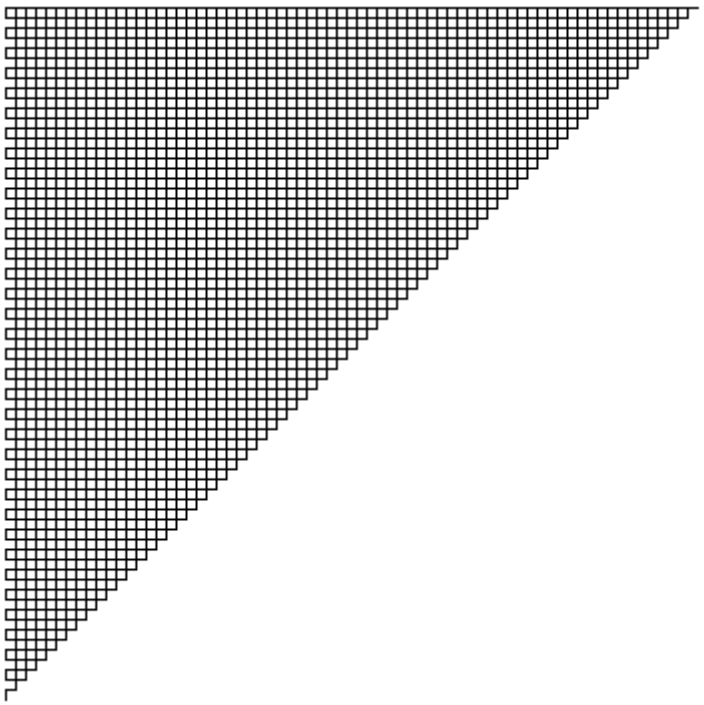


Figura 3.2

Curva generada por la función b_n definida a partir de la secuencia Golay-Rudin-Shapiro en el Ejemplo 3.0.3.

La Definición 3.0.1 que se mostró anteriormente es una definición robusta de las secuencias automáticas, pero si se consideran algunas variaciones simples, se puede ver que estas no cambian la clase de secuencias que consideramos automáticas. En la definición anterior se pide que $f_{\mathbb{M}}(u) = a_n$, con $\mathbb{M}$ un k-AFDS y para todo $u \in {\Sigma_k}^*$ tal que $[u]_k = n$, lo que incluye las entradas con ceros a la izquierda. El siguiente teorema muestra que de hecho, basta con que el k-AFDS $\mathbb{M}$ calcule la salida correcta sólo para las representaciones canónicas de n en base-k, es decir, aquellas que carecen de ceros a la izquierda.

Teorema 3.0.4. *La secuencia $(a_n)_{n\geq 0}$ es k-automática si y sólo si existe un k-AFDS $\mathbb{M}$ tal que $a_n = \tau(\delta(q_0,(n)_k))$, para todo $n \geq 0$, además, se puede elegir a $\mathbb{M}$ de tal forma que $\delta(q_0,0) = q_0$.*

Demostración: $\Rightarrow$) : *Es trivial, directo de la Definición 3.0.1.*

$\Leftarrow$) : *Considerando el k-AFDS $\mathbb{M} = (Q,\Sigma_k,\delta,q_0,\Delta,\tau)$, entonces definiendo el k-AFDS $\mathbb{M}' = (Q',\Sigma_k,\delta',q_0',\Delta,\tau')$ donde $Q' = Q \cup \{q_0'\}$, la función de transición $\delta' : Q' \times \Sigma_k \longrightarrow Q'$ definida para todo $q \in Q'$ y $a \in \Sigma_k$ como:*

$$\delta'(q,a) = \begin{cases} \delta(q,a) \ \ si\ q \neq q_0'\ y\ a \neq 0; \\ \delta(q_0,a) \ \ si\ q = q_0'\ y\ a \neq 0; \\ q_0' \ \ si\ q = q_0'\ y\ a = 0. \end{cases}$$

La función de salida $\tau' : Q' \longrightarrow \Delta$ definida para todo $q \in Q'$ como

$$\tau'(q) = \begin{cases} \tau(q) \ \ si\ q \neq q_0'; \\ \tau(q_0) \ \ si\ q = q_0'. \end{cases}$$

Es fácil ver que para todo $i \geq 0$ se tiene que

$$\tau'(\delta'(q_0',0^i(n)_k)) = \tau(\delta(q_0,(n)_k)).$$

Además, para $n \neq 0$, se tiene que $\delta'(q_0',0^i(n)_k) = \delta'(q_0',(n)_k) = \delta(q_0,(n)_k)$. ■

En la Sección 1.5 se presentó la Observación 1.5.3 donde se mencionó que por convención las representaciones se dan comenzando con el dígito más significativo, sin embargo haciendo una variación a esta condición en la definición de secuencia *k*-automática dada en la Definición 3.0.1 y con esto permitir que el AFDS procese de entrada la representación de *n* en base-*k* comenzando por el dígito menos significativo, de acuerdo a esto se tiene el siguiente resultado:

Teorema 3.0.5. *Las siguientes tres condiciones son equivalentes:*

(1) $(a_n)_{n\geq 0}$ es una secuencia k-automática.

(2) Existe un k-AFDS $\mathbb{M} = (Q,\Sigma_k,\delta,q_0,\Delta,\tau)$ tal que $a_n = \tau(\delta(q_0,u^R))$ para todo $n \geq 0$ y todo $u \in {\Sigma_k}^*$ tal que $[u]_k = n$.

(3) Existe un k-AFDS $\mathbb{M}' = (Q', \Sigma_k, \delta', q_0', \Delta, \tau')$ *tal que* $a_n = \tau'(\delta'(q_0', ((n)_k)^R))$ *para todo* $n \geq 0$.

Demostración: $(1) \Rightarrow (2)$: *Sea* $(a_n)_{n\geq 0}$ *una secuencia k-automática, entonces en existe una función* $f_{\mathbb{M}}$, *con un k-AFDS* $\mathbb{M}$ *tal que* $a_n = f_{\mathbb{M}}((n)_k)$. *Por el Teorema 2.4.7, la función* ${f_{\mathbb{M}}}^R(u) = f_{\mathbb{M}}(u^R)$ *que asigna la cadena* u^R *a* $f_{\mathbb{M}}((u^R)^R) = f_{\mathbb{M}}(u) = a_{[u]_k}$ *es una función de estados finitos, entonces existe un k-AFDS que la computa.*

$(2) \Rightarrow (3)$: *Es trivial haciendo* $u^R = ((n)_k)^R$.

$(3) \Rightarrow (1)$: *La condición* (3) *implica que existe una función de estados finitos* $g_{\mathbb{M}'}$ *tal que* $g_{\mathbb{M}'}(((n)_k)^R) = a_n$, *entonces por el Teorema 2.4.7, la función* ${g_{\mathbb{M}'}}^R$ *es una función de estados finitos tal que* ${g_{\mathbb{M}'}}^R((n)_k) = g_{\mathbb{M}'}(((n)_k)^R) = a_n$ *para todo* $n \geq 0$ *entero y así existe un k-AFDS que computa* ${g_{\mathbb{M}'}}^R$, *entonces por la Definición 3.0.1 implica que* $(a_n)_{n\geq 0}$ *es k-automática.* ■

Existe una segunda caracterización alterna de las secuencias automáticas a partir de k-fibras que se definen a continuación.

Definición 3.0.6 (La k-fibra de un símbolo en una secuencia)**.** Sea $u = (a_n)_{n\geq 0}$ una secuencia sobre un alfabeto Δ, un entero $k \geq 2$ y un símbolo $b \in \Delta$, entonces la k**-fibra del símbolo** b **en la secuencia** u, denotado por $I_k(u,b)$ es el conjunto

$$I_k(u,b) = \{(n)_k \in {\Sigma_k}^*, \text{ tal que } a_n = b\}.$$

Lema 3.0.7*. *Sea* Δ *un alfabeto y* $w = (a_n)_{n\geq 0}$ *una secuencia sobre* Δ. *La secuencia w es k-automática si y sólo si, cada fibra* $I_k(w,b)$ *es un lenguaje regular para todo* $b \in \Delta$.

Demostración: $\Rightarrow$) : *Sea* $w = (a_n)_{n\geq 0}$ *una secuencia k-automática sobre* Δ, *entonces existe un k-AFDS* $\mathbb{M} = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ *tal que* $a_n = f_{\mathbb{M}}((n)_k) = \tau(\delta(q_0, (n)_k))$ *para todo* $n \geq 0$ *entero. Considerando el lenguaje regular* $L_k(b)$, *para todo* $b \in \Delta$, *definido como:*

$$L_k(b) = \{u \in {\Sigma_k}^* : \tau(\delta(q_0, u)) = b\}.$$

El lenguaje $L_k(b)$ *es regular, para todo* $b \in \Delta$, *ya que es la unión de lenguajes regulares de la*

forma $\{u \in {\Sigma_k}^*,$ *tal que* $\delta(q_0,u) = q\}$ *para todo* $q \in Q$ *tal que* $\tau(q) = b$*, esto es:*

$$L_k(b) = \bigcup_{\{q \in Q,\ tal\ que\ \tau(q)=b\}} \{u \in {\Sigma_k}^*,\ tal\ que\ \delta(q_0,u) = q\}.$$

Los conjuntos de la forma $\{u \in {\Sigma_k}^*,$ *tal que* $\delta(q_0,u) = q\}$ *para todo* $q \in Q$ *tal que* $\tau(q) = b$*, son lenguajes regulares sobre* Σ^**, ya que para cada uno existe un AFD* $\mathbb{M}_b{}' = \{Q, \Sigma_k, \delta, q_0, F\}$*, donde* Q*,* δ *y* q_0 *son iguales que en* $\mathbb{M}$ *y el conjunto de estados finales es* $F = \{q \in Q,$ *tal que* $\tau(q) = b\}$*, tal que* $\mathscr{L}(\mathbb{M}_b{}') = \{u \in {\Sigma_k}^*,$ *tal que* $\delta(q_0,u) = q\}$ *para todo* $q \in Q$ *tal que* $\tau(q) = b$*. Al final del Corolario 1.5.2 se definió el lenguaje regular* C_k*, luego,* $L_k(b) \cap C_k$ *es un lenguaje regular, para todo* $q \in \Delta$*, por el Teorema 2.1.8 y* $L_k(b) \cap C_k = I_k(w,b)$*, para todo* $q \in \Delta$*.*

$\Leftarrow$) : *Si* $I_k(w,b) = \{(n)_k \in {\Sigma_k}^*,$ *tal que* $a_n = b\}$ *es regular para todo* $b \in \Delta$*, sin pérdida de generalidad suponiendo que* $\Delta = \{1,2,\ldots,r\}$*. Considerando el conjunto* ${I_k}'(b)$ *que es un lenguaje regular sobre* ${\Sigma_k}^*$*, tal que para todo* $b \in \Delta$ *se define como:*

$$I'_k(b) := 0^* I_k(w,b).$$

Puesto que, ${I_k}'(b)$ *es un lenguaje regular para todo* $b \in \Delta$ *y* ${I_k}'(b) \cap {I_k}'(c) = \emptyset$ *para* b *y* $c \in \Delta$ *tal que* $b \neq c$*, entonces como*

$$ {\Sigma_k}^* = \bigcup_{\{b \in \Delta\}} {I_k}'(b),$$

es una partición de ${\Sigma_k}^*$ *y por el Teorema 2.4.6, existe un AFDS* $\mathbb{M} = \{Q,\ \Sigma_k, \delta,\ q_0,\ \Delta,\ \tau\}$*, tal que para toda cadena* $u \in {\Sigma_k}^*$ *con* $[u]_k = n$ *y para todo* $n \geq 0$ *entero se tiene que:*

$$\begin{aligned} \tau(\delta(q_0,u)) = b \in \Delta &\iff u \in {I_k}'(b); \\ &\iff u = vu' \ tal\ que\ v \in 0^*\ y\ u' \in I_k(w,b); \\ &\iff [u]_k = [vu']_k = [u']_k\ y\ a_{[u]_k} = b; \\ &\iff b = a_n. \end{aligned}$$

Finalmente por el Teorema 3.0.4, la secuencia $w = (a_n)_{n \geq 0}$ *sobre* Σ_k *es k-automática.*■

Los siguientes teoremas enuncian algunos resultados importantes sobre secuencias automáticas.

Teorema 3.0.8. *Si una secuencia $w = (a_n)_{n\geq 0}$ sobre un alfabeto Δ difiere sólo en un número finito de términos de una secuencia k-automática $w' = (b_n)_{n\geq 0}$ sobre Δ, entonces la secuencia w es k-automática.*

Demostración: Por hipótesis existe $i \geq 0$ tal que $a_n = b_n$ para $n \geq i$. Sea $v = w[0..i-1] \in \Delta^* = (a_n)_{n>i}$ y $u = (a_n)_{n\geq i}$, entonces u es k-automática, es decir, toda fibra $I_k(u,b)$ es un lenguaje regular para todo $b \in \Delta$. Faltaría probar que $I_k(v,b)$ es un lenguaje regular para $v = w[0..i-1]$ y $b \in \Delta$. Esto es ya que el conjunto $I_k(v,b) = \{(n)_k \in \Sigma^*$, tal que $v[n] = b\}$, como v es una cadena finita, entonces los conjuntos $I_k(v,b)$ son lenguajes finitos sobre el alfabeto Σ^* para todo $b \in \Delta$, aplicando el Teorema 1.3.20, el cual dice que todo lenguaje finito es regular, se tiene que $I_k(v,b)$ es regular para todo $b \in \Delta$. Así por el Lema 3.0.7, entonces la secuencia $w = vu = w[0..i-1](a_n)_{n\geq i}$ es k-automática. ■

En la Definición 1.2.14 se introdujeron las cadenas finalmente periódicas, para cadenas infinitas sobre un alfabeto que son de la forma $u = wv^w$, este concepto se aplica de igual forma a las secuencias, debido a que pueden definirse como una cadena infinita a derecha. A continuación un resultado que relaciona esa propiedad con las secuencias automáticas.

Teorema 3.0.9. *Sea Δ un alfabeto. Si $(a_n)_{n\geq 0}$ es una secuencia finalmente periódica sobre Δ, entonces es k-automática para todo $k \geq 2$.*

Demostración: Como $u = (a_n)_{n\geq 0}$ es una secuencia finalmente periódica, entonces $u = wv^w$, con w y v palabras finitas sobre Σ, a partir del Teorema 3.0.8, bastaría probar que v^w es k-automática. Suponiendo entonces que $v^w = vvv\ldots = (b_n)_{n\geq 0}$ con $v \in \Sigma$ tal que $|v| = t$, esto es que $b_{tn+i} = b_i$ para $0 \leq i < t$ y $n \geq 0$. Considerando el autómata $\mathbb{M} = (Q, \Sigma, \delta, q_0, \Delta, \tau)$, donde $Q = \Sigma_t = \{0,1,\ldots,t-1\}$ es el conjunto de estados, $\Sigma = \Sigma_k = \{0,1,\ldots,k-1\}$ el alfabeto de entrada, la función de transición $\delta : \Sigma_t \times \Sigma_k \longrightarrow \Sigma_t$ es definida para todo $q \in Q$ y $b \in \Sigma_k$ como: $\delta(q,b) = (kq+b) \pmod t$. El estado inicial $q_0 = 0$ y la función de salida $\tau : Q \longrightarrow \Delta$ es definida para todo $q \in Q$ como: $\tau(q) = b_q$. Sea $u = b_0 b_1 \cdots b_{j-1} \in {\Sigma_k}^*$, se probará por inducción sobre $|u| = j \geq 1$ que $\delta(q_0,u) = \delta(q_0, b_0 b_1 \cdots b_{j-1}) = ([b_0 b_1 \cdots b_{j-1}]_k) \pmod t$.

(i) Si $|u| = 1$, entonces $u = b_0 \in \Sigma_k$, es decir, $0 \leq b_0 < k$ entero y entonces:

$$\begin{aligned}\delta(q_0,u) &= \delta(q_0,b_0);\\ &= k\times 0+b_0 \pmod t;\\ &= b_0\times k^0 \pmod t;\\ &= ([b_0]_k) \pmod t.\end{aligned}$$

(ii) Suponiendo que para $|u| = j > 1$ *se cumple que* $\delta(q_0,u) = \delta(q_0,b_0b_1\cdots b_{j-1}) = ([b_0b_1\cdots b_{j-1}]_k) \pmod t$*, hay que probar que para* $|u| = j+1$ *también se tiene que* $\delta(q_0,u) = \delta(q_0,b_0b_1\cdots b_j) = ([b_0b_1\cdots b_j]_k) \pmod t$.

Dado que:

$$\begin{aligned}\delta(q_0,b_0b_1\cdots b_j) &= \delta(\delta(q_0,b_0b_1\cdots b_{j-1}),b_j));\\ &= \delta((([b_0b_1\cdots b_{j-1}]_k) \pmod t),b_j);\\ &= (k\times ((([b_0b_1\cdots b_{j-1}]_k) \pmod t))+b_j) \pmod t;\\ &= (k\times (b_0k^j \pmod t)+\cdots+b_{j-1}k^0 \pmod t))+b_j) \pmod t;\\ &= (b_0k^{j+1} \pmod t)+\cdots+b_{j-1}k^1 \pmod t)+bjk^0 \pmod t)) \pmod t;\\ &= ([b_0b_1\cdots b_j]_k) \pmod t.\end{aligned}$$

Finalmente, se probará que $f_{\mathbb{M}}((n)_k) = b_n \in \Delta$*, para todo* $n \geq 0$*. En efecto, sea* $n \geq 0$*, entonces por el algoritmo de la división* $n = t\times k+i$*, con* $k \geq 0$ *y* $0 \leq i < t$*, es decir,* $i \equiv n \pmod t$*, así se tiene que:*

$$\begin{aligned}f_{\mathbb{M}}((n)_k) &= \tau(\delta(q_0,(n)_k));\\ &= \tau([(n)_k]_k \pmod t));\\ &= \tau(n \pmod t)) = \tau(i \pmod t)),\ \textit{con}\ 0 \leq i < t;\\ &= \tau(i) = b_i = b_n.\end{aligned}$$

Así, $(b_n)_{n\geq 0}$ *es k-automática y por tanto, del Teorema 3.0.8,* $(a_n)_{n\geq 0}$ *también lo es.* ■

Teorema 3.0.10. *Sea* Δ *un alfabeto,* $u = (a_n)_{n\geq 0}$ *una secuencia k-automática sobre* Δ *y sea* $\varphi : \Delta \longrightarrow \Delta'$ *una codificación, entonces la secuencia* $\varphi(u)$ *también es k-automática.*

Demostración: Puesto que u es k-automática, existe un k-AFDS $\mathbb{M} = (Q,\Sigma,\delta,q_0,\Delta,\tau)$ *tal que* $\tau(\delta(q_0,(n)_k)) = a_n$ *para todo* $n \geq 0$*. Considerando el k-AFDS* $\mathbb{M}' = (Q,\Sigma,\delta,q_0,\Delta,\varphi\circ\tau)$ *con* $\varphi\circ\tau$ *la composición de la función de salida* τ *de* $\mathbb{M}$ *y* φ *el homomorfismo de cadenas, entonces se tiene que* $\varphi(\tau(\delta(q_0,(n)_k))) = \varphi(a_n)$*, para todo* $n \geq 0$*.* ■

Teorema 3.0.11. *Sean Δ' y Δ'' dos alfabetos y $u = (a_n)_{n\geq 0}$ y $v = (b_n)_{n\geq 0}$ dos secuencias k-automáticas sobre Δ' y Δ'' respectivamente, entonces la secuencia $u \times v = ((a_n, b_n))_{n\geq 0}$ es k-automática.*

Demostración: Como u y v son k-automáticas, existen dos k-AFDS $\mathbb{M} = (Q', \Sigma_k, \delta', q_0', \Delta', \tau')$ y $\mathbb{M}' = (Q'', \Sigma_k, \delta'', q_0'', \Delta'', \tau'')$ que generan a u y v respectivamente. Considerando el k-AFDS $\mathbb{M}'' = (Q = Q' \times Q'', \Sigma_k, \delta, q_0 = (q_0', q_0''), \Delta = \Delta' \times \Delta'', \tau)$ donde la función de transición $\delta : Q \times \Sigma_k \longrightarrow Q$ es definida para todo $q = (q', q'') \in Q$ y $a \in \Sigma_k$ como: $\delta(q, a) = (\delta'(q', a), \delta''(q'', a))$ y la función de salida $\tau : Q \longrightarrow \Delta$ es definida para todo $q = (q', q'') \in Q$ como: $\tau(q) = (\tau'(q'), \tau''(q''))$, entonces se tiene que $\tau(\delta(q_0, (n)_k)) = (a_n, b_n)$, para todo $n \geq 0$. ■

Por consecuencia de los Teoremas 3.0.10 y 3.0.11, se tiene el siguiente resultado.

Corolario 3.0.12*. *Sean Δ_1, Δ_2 y Δ_3 tres alfabetos, $u = (a_n)_{n\geq 0}$ y $v = (b_n)_{n\geq 0}$ dos secuencias k-automáticas sobre Δ_1 y Δ_2 respectivamente y $f : \Delta_1 \times \Delta_2 \longrightarrow \Delta_3$ cualquier función. Entonces la secuencia $(f(a_n, b_n))_{n\geq 0}$ es k-automática.*

3.1. Conjuntos automáticos

En esta sección se da una pequeña mención de los denominados conjuntos k-automáticos y una definición acorde a los resultados anteriormente enunciados y la mención de las propiedades más importantes.

Definición 3.1.1 (Conjuntos k-automáticos)**.** Sea $S \subseteq \mathbb{N}$ un conjunto de enteros no negativos, entonces S se denomina k-automático si su función característica $f_S : \mathbb{N} \longrightarrow \Sigma_2$, genera una secuencia k-automática de la forma $(f_s(n))_{n\geq 0}$.

De otra forma, la secuencia característica de S, $u = (a_n)_{n\geq 0}$, la cual cumple que $a_n = f_S(n)$ para todo $n \geq 0$, se dice que es k-automático si la secuencia u es k-automática.

Ejemplo 3.1.2. Sea S el conjunto de potencias de 2, $S = \{1, 2, 4, 8, 16, \ldots\}$, S es un conjunto 2-automático ya que existe una función de estados finitos sobre el alfabeto Σ_2: $f : {\Sigma_2}^* \longrightarrow \Sigma_2$ para todo $n \in \mathbb{N}$ y $f((n)_2) = f_S(n)$. Considerando el autómata k-AFDS $\mathbb{M} = \{Q = \{q_0, q_1, q_2\}, \Sigma_2, \delta, q_0, \Sigma_2, \tau\}$, donde la función de transición $\delta : Q \times \Sigma_2 \longrightarrow Q$ es

ilustrada en la Tabla 3.4, y la función de salida $\tau : Q \longrightarrow \Sigma_2$ la Tabla 3.5. Finalmente en la Tabla 3.6 se muestra los primeros 10 términos de la sucesión $(f_s(n))_{n\geq 0}$.

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_2

Tabla 3.4

Tabulación de la función de transición δ en el Ejemplo 3.1.2.

q	$\tau(q)$
q_0	ε
q_1	1
q_2	0

Tabla 3.5

Tabulación de la función de salida τ en el Ejemplo 3.1.2.

n	0	1	2	3	4	5	6	7	8	9	10
$(n)_2$	0	1	10	11	100	101	110	111	1000	1001	1010
$\delta(q_0,(n)_2)$	q_0	q_1	q_1	q_2	q_1	q_2	q_2	q_2	q_1	q_2	q_2
$\tau(\delta(q_0,(n)_2) = f_S(n)$	ε	1	1	0	1	0	0	0	1	0	0

Tabla 3.6

Tabulación de los primeros diez términos de la secuencia $(f_s(n))_{n\geq 0}$ en el Ejemplo 3.1.2.

Ejemplo 3.1.3. Del Teorema 3.0.9, cualquier progresión aritmética de enteros es un conjunto 2-automático, pues toda progresión aritmética podría verse como una secuencia finalmente periódica.

En [3, pág 168 - 169] se muestra el siguiente teorema y su demostración para cada una de las operaciones, sin embargo en este documento se presentará como una proposición consecuente de muchos de los resultados anteriormente demostrados.

Proposición 3.1.4. Sea k fijo, la clase de conjuntos k-automáticos es cerrada bajo las operaciones:

(a) Intersección;

(b) Unión;

(c) Complemento;

(d) Conjunto suma, es decir, la operación $A+B=\{a+b : a\in A, b\in B\}$;

(e) Multiplicación con entero no negativo, es decir, la operación $nA=\{na : a\in A\}$;

(f) La operación definida por $J_{a,b}(S)=\{x : ax+b\in S\}$, donde $a\in\mathbb{N}$ y $b\in\mathbb{Z}$.

3.2. Teorema de Cobham

En en la Sección 1.4 se introdujeron los homomorfismos uniformes y a su vez la idea de la iteración de un homomorfismo a partir de un punto prolongable que permite generar cadenas infinitas. En esta sección se muestra el teorema que permite relacionar las secuencias generadas por un autómata k-automático y las secuencias generadas por la iteración de un homomorfismo uniforme y prolongable. Para ello es necesario mostrar inicialmente el siguiente lema.

Lema 3.2.1*. *Sea $u=a_0a_1a_2\cdots$ una palabra infinita a derecha en Σ^w, tal que $u=\varphi(u)$, con φ un homomorfismo k-uniforme, con $k\geq 0$ entero. Entonces $\varphi(a_i)=a_{(k\times i)}a_{(k\times i)+1}a_{(k\times i)+2}\cdots a_{(k\times i)+k-1}$.*

Demostración:

Como $u=\varphi(u)$ y φ es un homomorfismo k-uniforme, entonces:

$$\begin{aligned}\varphi(a_0a_1\cdots a_i) &= a_0a_1\cdots a_{(k\times i)+k-1};\\ &= (a_0a_1\cdots a_{(k\times i)-1})(a_{ki}a_{ki+1}\cdots a_{(k\times i)+k-1});\\ &= \varphi(a_0a_1\cdots a_{i-1})\varphi(a_i).\end{aligned}$$

Así, $\varphi(a_i)=a_{(k\times i)}a_{(k\times i)+1}\cdots a_{(k\times i)+k-1}$.

■

En otra notación, $\varphi(a_i)=u[(k\times i)\,..\,((k\times i)+k-1)]$,[3] *donde $u[(k\times i)\,..\,((k\times i)+k-1)]$ denota la cadena de símbolos desde la posición $(k\times i)$ hasta la posición $((k\times i)+k-1)$, es decir $a_{(k\times i)}a_{(k\times i)+1}\cdots a_{(k\times i)+k-1}$.*

[3]Esta notación fue introducida en la Notación 1.1.10.

Se tienen entonces las herramientas necesarias para mostrar el siguiente teorema, el cual constituye el resultado principal de este documento.

Teorema 3.2.2 (Teorema de Cobham ★)**.** *Sean Δ un alfabeto finito y una secuencia $u = (a_n)_{n\geq 0}$ sobre Δ, entonces u es k-automática si y sólo si existen, un alfabeto Γ, un homomorfismo k-uniforme $\varphi : \Gamma^* \longrightarrow \Gamma^*$ que es prolongable en algún $a \in \Gamma$ y una codificación $\tau : \Gamma^* \longrightarrow \Delta^*$ tal que $\tau(\varphi^w(a)) = u$.*

Demostración: $\Rightarrow$) : Como u es una secuencia k-automática, con $k \geq 0$ entero, entonces existe un k-AFDS $\mathbb{M} = (Q,\ \Sigma_k,\ \delta,\ q_0,\ \Delta,\ \tau)$, tal que $\tau(\delta(q_0,(n)_k)) = a_n$, para todo $n \geq 0$. Entonces el homomorfismo $\varphi : \Gamma^* \longrightarrow \Gamma^*$ con $\Gamma = Q$ se define de la siguiente manera para todo $q \in Q$;

$$\varphi(q) = \delta(q,0)\delta(q,1)\cdots\delta(q,k-1). \tag{3.3}$$

Por el Teorema 3.0.4, sin pérdida de generalidad, se asume que $\delta(q_0,0) = q_0$. Considerando el punto fijo de φ definido como $w =: \varphi^w(q_0)$,[4] hay que probar por inducción sobre $|u|$, para toda cadena $u \in \Sigma^*$, que:

$$\delta(q_0,u) = w[[u]_k]. \tag{3.4}$$

Donde $w[i]$ es el símbolo de w en la posición i,[5] para $i \geq 0$ entero.
Se tiene que:

(i) Para $|u| = 0$, $\delta(q_0,\varepsilon) = q_0 = w[0] = w[[\varepsilon]_k]$.

(ii) Suponiendo que la ecuación (3.4) se cumple para todo $u \in {\Sigma_k}^*$ con $|u| \leq i-1$, hay que probar que también se cumple para todo $u \in {\Sigma_k}^*$ con $|u| = i$. En efecto, esto es porque: Dado que $|u| = i$, u se puede descomponer como $u = va$, con $v \in {\Sigma_k}^*$ y $a \in \Sigma_k$, tal que $|v| = i-1$. Entonces:

$$\begin{aligned}
\delta(q_0,u) = \delta(q_0,va) &= \delta(\delta(q_0,v),a);\\
&= \delta(w[[v]_k],a),\ \textit{por el paso inductivo y } |v| = i-1;\\
&= (\varphi(w[[v]_k]))[a],\ \textit{tomando la posición a de la ecuación (3.3) evaluada en } (w[[v]_k]);\\
&= (w[(k\times[v]_k)\ ..\ ((k\times[v]_k)+k-1)])[a],\ \textit{por el Lema 3.2.1};\\
&= w[((k\times[v]_k)+a)] = w[[va]_k];\\
&= w[[u]_k].
\end{aligned}$$

Como se quería.

Por lo anterior y de la hipótesis, entonces:

$$\tau(w[n]) = \tau(w[[(n)_k]_k]) = \tau(\delta(q_0,(n)_k)) = a_n.$$

[4]Esta notación fue introducida en la Observación 1.4.10.
[5]Esta notación fue introducida en la Notación 1.1.10.

Finalmente, se tiene entonces que,

$$\tau(w) = \tau(\varphi^w(q_0)) = a_0a_1a_2\cdots = (a_n)_{n\geq 0} = u.$$

$\Leftarrow$) : *Suponiendo que* Γ *y* Δ *son dos alfabetos finitos,* $u = a_0a_1a_2\cdots = (a_n)_{n\geq 0} = \tau(w)$, *donde* $w := \varphi^w(a)$, *para algún homomorfismo k-uniforme* $\varphi : \Gamma^* \longrightarrow \Gamma^*$ *prolongable en* $a \in \Gamma$ *y* $\tau : \Gamma^* \longrightarrow \Delta^*$ *es una codificación. Considerando el k-AFDS* $\mathbb{M} = (Q,\ \Sigma_k,\ \Delta,\ \delta,\ q_0,\ \tau)$, *donde* $Q = \Gamma$, $q_0 = a$ *y la función de transición* $\delta : Q \times \Sigma_k \longrightarrow Q$ *es definida para todo* $q \in \Gamma$ *y* $c \in \Sigma_k$ *como:*

$$\delta(q,c) := (\varphi(q))[c]. \tag{3.5}$$

Hay que probar por inducción sobre $n \geq 0$ *que:*

$$\delta(q_0,(n)_k) = w[n]. \tag{3.6}$$

Esto es, porque:

(i) Si $n = 0$, $\delta(q_0,(0)_k) = \delta(q_0,\varepsilon) = q_0 = a = w[0]$.

(ii) Suponiendo que la ecuación (3.6) se cumple para todo $n \leq i-1$, *con* $i \geq 0$ *un entero. Hay que probar que la ecuación (3.6) se cumple para* $n = i$. *En efecto,* $(n)_k$ *se puede descomponer como* $(n)_k = va$, *con* $v \in {\Sigma_k}^*$ *y* $a \in \Sigma_k$, *entonces* $n = k \times n' + a$, *con* $n' = [v]_k < n$ *y* $(n')_k = v$.

Entonces:

$$\begin{aligned}
\delta(q_0,(n)_k) &= \delta(q_0,va) = \delta(\delta(q_0,v),a);\\
&= \delta(\delta(q_0,(n')_k),a);\\
&= \delta(w[n'],a),\ \textit{por el paso inductivo y } n' < i;\\
&= (\varphi(w[n']))[a],\ \textit{por la ecuación (3.5)};\\
&= (w[(k\times n').. ((k\times n')+k-1)])[a],\ \textit{por el Lema 3.2.1};\\
&= w[k\times n' + a];\\
&= w[n].
\end{aligned}$$

De esta forma se tiene que $\tau(\delta(q_0,(n)_k)) = \tau(w[n]) = u[n] = a_n$.

■

A partir de este teorema se cuenta con una forma trivial de convertir un k-AFDS que computa una secuencia automática, en el sentido que se definió anteriormente, en la representación correspondiente de la forma de homomorfismo $\tau(\varphi^w(u))$ y viceversa, usando las correspondencias:

(i) Símbolos (letras) $\longleftrightarrow$ estados.

(ii) Imágenes de los símbolos bajo el homomorfismo $\varphi \longleftrightarrow$ función de transición entre estados con entradas en $\Sigma_k = \{0,1,\ldots,k-1\}$.

(iii) Imágenes de la función $\tau \longleftrightarrow$ Salidas de los estados.[6]

[6]Esta demostración fue estudiada de [3, pág 175 - 176] y de [4, pág 65 - 68].

3.2.3 Observación. Para cada secuencia k-automática u, existe un único k-AFDS con el menor o mínimo número de estados, para $k \geq 0$. (La unicidad depende de la denominación o notación de los estados), suponiendo que son denotados de la forma $q_0, q_1, \ldots,$, entonces existe una única forma de denotar los estados, de la siguiente manera: q_0 es el estado inicial, q_1 es el primer estado de la forma $\delta(q_0, i)$ que no es igual a q_0, q_2 es el primer estado de la forma $\delta(q_0, j)$ que no es igual a q_0 o a q_1, si no existe tal $\delta(q_0, j)$, entonces q_2 es el primer estado de la forma $\delta(q_1, i)$ diferente de q_0, q_1 y así sucesivamente para cada estado q_k.
Esto significa que, si u es una secuencia k-automática, para $k \geq 0$, entonces existe una cadena canónica o única w en el conjunto de cadenas infinitas a derecha, ${\Sigma_k}^w$ y una codificación τ tal que $u = \tau(w)$ y la secuencia w es denominada **secuencia interior** de u.

Ejemplo 3.2.4. A partir del AFDS considerado en el Ejemplo 3.0.3, se puede construir un homomorfismo φ tal que $\varphi(q_{00}) = q_{00}q_{01},\ \varphi(q_{01}) = q_{00}q_{11},\ \varphi(q_{10}) = q_{10}q_{11},\ \varphi(q_{11}) = q_{10}q_{01}$ y el codificador τ tal que $\tau(q_{00}) = 1,\ \tau(q_{01}) = 1,\ \tau(q_{10}) = -1,\ \tau(q_{11}) = -1$. La secuencia Golay-Rudin-Shapiro definida en el Ejemplo 3.0.3 puede ser generada como $\tau(\varphi^w(q_{00}))$.

Ejemplo 3.2.5. Sea $\Sigma = \{a, b, c, d\}$ y $\varphi : \Sigma^* \longrightarrow \Sigma^*$ un homomorfismo uniforme tal que:

$$\begin{aligned} \varphi(a) &= bc, & \varphi(b) &= cd; \\ \varphi(c) &= da, & \varphi(d) &= ab. \end{aligned}$$

En la Figura 3.3 se muestra el 2-AFDS que genera la secuencia automática $\varphi^w(a)$.

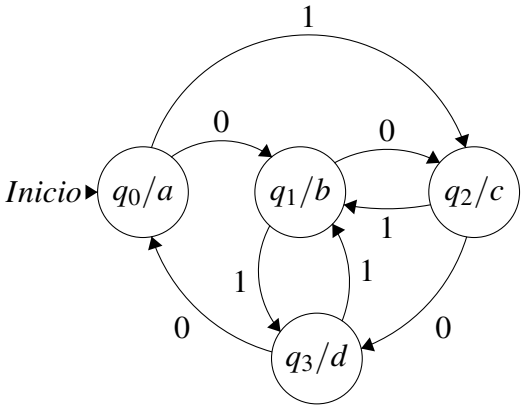


Figura 3.3

Diagrama de transición del 2-AFDS que genera la secuencia en el Ejemplo 3.2.5.

4. CURVAS GENERADAS A PARTIR DE SECUENCIAS AUTOMÁTICAS

En este último capítulo se expone un puente entre el teorema de Cobham anteriormente mencionado, enmarcado en la teoría computacional y los fractales o curvas generadas a partir de las interpretaciones geométricas o en este caso, a partir del lenguaje LOGO de los L-sistemas que también pueden ser secuencias automáticas y por ende permiten definir y asociar diferentes reglas de dibujo análogas a las usadas por el biólogo Aristid Lindenmayer en los inicios de esta teoría. Para el desarrollo de este capítulo se ha tomado como referencia [3] [4], [11], [12], [13], [14] y [15].

4.1. Gramáticas generativas

Las gramáticas generativas fueron introducidas por Noam Chomsky en 1956 como un modelo para describir lenguajes naturales como el español y el inglés. Chomsky clasificó las gramáticas en cuatro tipos y a cada uno le asignó un dígito desde 0 al 3, de acuerdo a las propiedades de sus reglas de producción y las condiciones en las que pueden ser aplicadas a una cadena en cada paso de la producción. Cabe resaltar que de estas clasificaciones, las gramáticas de tipo 2, llamadas gramáticas independientes del contexto, se comenzaron a utilizar desde los años setenta para presentar la sintaxis de lenguaje de programación y entre otras aplicaciones importantes. Se hace una pequeña introducción a la definición de gramáticas generativas para poder remarcar y mostrar la diferencia entre éstas y los L-sistemas que serán presentados más adelante.

Definición 4.1.1. Una **gramática generativa** es definida como una cuádrupla

$$\mathbb{G} = (V, \Sigma, S, P), \tag{4.1}$$

donde,

1. V es el alfabeto de símbolos no-terminales.
2. Σ es el alfabeto de símbolos terminales, con $\Sigma \cap V = \emptyset$.
3. $S \in V$, denominado símbolo inicial.
4. P es un conjunto finito de producciones o reglas de re-escritura con $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$.

Usualmente los símbolos no-terminales son representados por letras mayúsculas y los símbolos terminales con letras minúsculas. Los elementos de P, el conjunto de producciones, de la forma (v,w) son denotados por $v \to w$ y significa que la cadena v se puede reemplazar (sobre-escribir) por la cadena w. Comenzando con el símbolo inicial S y aplicando una de las reglas de producciones de la gramática G en cada paso y se generan cadenas de terminales y no-terminales. Las cadenas que se generan a partir de estas reglas de producciones y que sólo contienen símbolos terminales conforman el lenguaje generado por G denotado por $\mathscr{L}(G)$.

Ejemplo 4.1.2. Sea $G = (V = \{S,A,B\}, \Sigma = \{a,b\}, S, P)$ la gramática dada por las en P:

$$\begin{array}{lll} S \to ABS, & S \to \varepsilon, & BA \to AB; \\ BS \to b, & Bb \to bb, & Ab \to ab; \\ Aa \to aa. & & \end{array}$$

La gramática G define el lenguaje de todas las cadenas de la forma $a^n b^n$, es decir $\mathscr{L}(G) = \{a^n b^n : n \in \mathbb{N}\}$. Para $n = 1$, la cadena $a^n b^n = ab$ es obtenida de la siguiente forma:

$$\begin{array}{rl} S \to & ABS, \quad \text{puesto que } S \to ABS; \\ ABS \to & Ab, \quad \text{dado que } BS \to b; \\ Ab \to & ab, \quad \text{porque } Ab \to ab. \end{array}$$

Para $n = 2$, la cadena $a^n b^n = aabb$ es obtenida de la siguiente forma:

$$\begin{array}{rll} S \to ABS \to & ABABS, & (S \to ABS); \\ & ABABS \to AABBS, & (BA \to AB); \\ & AABBS \to AABb, & (BS \to b); \\ & AABb \to AAbb, & (Bb \to bb); \\ & AAbb \to Aabb, & (Ab \to ab); \\ & Aabb \to aabb, & (Aa \to aa); \end{array}$$

Ejemplo 4.1.3. Sea $G' = (V = \{S\}, \Sigma = \{a,b\}, S, P)$ la gramática dada por la siguientes producciones en P:

$$S \to aSb \text{ y } S \to \varepsilon.$$

Al igual que el ejemplo anterior las producciones de G' define el lenguaje de todas las cadenas de la forma $a^n b^n$, es decir $\mathscr{L}(G') = \{a^n b^n : n \in \mathbb{N}\}$.

En la Sección 1.3 se presentó una introducción de los lenguajes regulares y en el segundo capítulo se mostró una caracterización de esos lenguajes a partir de los autómatas, en el Teorema 2.3.1 (Teorema de Kleene), el cual enuncia que un lenguaje regular es precisamente aquel que es aceptado por alguna clase de autómata de estados finitos. Finalizando el segundo capítulo se definieron los autómatas finitos deterministas con salida y se mostró que esta clase de autómatas generan cierto lenguaje que en primera instancia también es reconocido por un autómata de estados finitos (AFD - AFND - AFND-ε), por otro lado, las gramáticas generativas son otro método que nos permite generar lenguajes de diferentes categorías y clasificarlos según el tipo de gramática que lo genere. Finalmente en el tercer capítulo se mostró la relación entre las secuencias automáticas y su generación por medio de homomorfismos uniformes. Esta propiedad de las secuencias automáticas permite relacionarlas con un tipo de gramática diferentes a las gramáticas generativas mostradas anteriormente y que se conocen como **L-sistemas**, los cuales se definen en la siguiente sección.

4.2. L-Sistemas

Los L-Sistemas o también conocidos como sistemas de reescritura de cadenas son un nuevo tipo de gramática (gramática de derivación paralela), frente a las gramáticas generativas y formales, los cuales fueron introducidos por el biólogo Aristid Lindenmayer en 1986 en los artículos [16] y [17]; interesado por desarrollo biológico de los organismos filamentosos, Lindenmayer fue pionero en lo que hoy en día se conoce como sistemas de reescritura de palabras, usada por él para describir una base de una teoría axiomática del desarrollo biológico. El método definido por Lindenmayer constituye un mecanismo alterno a las sistemas iterados de funciones para la generación de fractales y curvas, como se verá más adelante. Además, una diferencia principal entre las gramáticas de Chomsky (gramáticas generativas y formales) y los L-sistemas es el método de aplicación de producciones, puesto que en las gramáticas de Chomsky, las producciones se aplican secuencialmente, mientras que en los L-sistemas, las producciones se aplican en paralelo, es decir, reemplazando simultáneamente todas las

producciones posibles de una cadena dada en cada paso. Al igual que las gramáticas generativas, existe también una clasificación de los L-sistemas. Sin embargo, los únicos L-sistemas que serán definidos y empleados aquí son denominados **D0L-Sistemas** (O por sus siglas, Sistemas deterministas y 0-contexto o de contexto libre).

Definición 4.2.1. Un **D0L-sistema** es una tripleta

$$G = (\Sigma, \varphi, \omega), \tag{4.2}$$

donde,

1. Σ es un alfabeto finito.
2. φ es un homomorfismo $\varphi : \Sigma^* \longrightarrow \Sigma^*$.
3. $\omega \in \Sigma^+$ es denominada axioma.

Dado un D0L-sistema G como en (4.2), entonces, para cada $a \in \Sigma$, existe $\varphi(a) = u \in {\Sigma_K}^*$, tal que el par (a,u) es llamado **regla de producción** y también puede denotares simplemente con $a \to u$. Además, el símbolo en la izquierda de la regla de producción es denominado **variable**. A partir de un D0L-sistema, es posible generar la secuencia $g_1 g_2 \cdots = (\varphi^n(\omega))_{n\geq 1}$ que es gráficamente interpretable. A continuación se presenta un ejemplo mostrado por Lindenmayer.

Ejemplo 4.2.2. Sea el D0L-sistema $G = (\Sigma = \{a,b\}, \varphi, \omega = b)$ con las reglas de producción $a \to ab$ y $b \to a$. En la Figura 4.1 se muestra las primeras 5 iteraciones de las reglas de producción de G sobre el axioma $\omega = b$.
Este L-sistema es conocido como el Fibonacci L-sistema[1] por su conexión con la secuencia que lleva el mismo nombre donde $F_0 = F_1 = 1$ y $F_{n+2} = F_{n+1} + F_n$ para todo $n \geq 0$.

En la Tabla 3.3 se muestran las primeras 8 generaciones y se puede notar la conexión con la secuencia de Fibonacci, pues para cada generación, recordando la notación presentada para las cadenas, se tiene que $|g_n| = F_n$, y por el patrón $g_{n+2} = g_{n+1} g_n$, que por notación de concatenación significa que los símbolos de la generación $n+2$ son los mismos símbolos de la

[1] Se puede pensar que las letras a y b representan formas de vida y las reglas de producción como etapas de su vida, por ejemplo, tras la primera etapa o aplicación de la regla de producción, se puede considerar que la forma de vida "inmadura" representada por la letra b (bebé) pasó a ser un adulto ahora representada por la letra a. Después el adulto a es capaz de producir un bebé b en cada generación y así este sistema dinámico de símbolos puede modelar uno de los modelos de dinámicas de población más simples que existen.

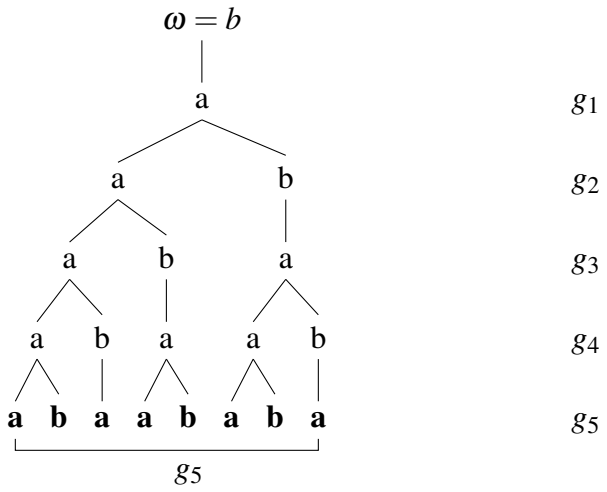


Figura 4.1

Fibonacci L-sistema.

generación $n+1$ concatenados con los símbolos de la generación n, por ejemplo:

$$g_8 = (abaabaabaabaabaabaabaaba)(abaabaabaabaaba) = g_7 g_6.$$

El proceso global y particular detrás de la evolución del sistema de Fibonacci es llamado **ley de cadena o concatenación**, puesto que se usan generaciones pasadas para concatenarse y formar la nueva generación. Formalmente, se dice que un L-sistema satisface una ley de cadena si existen una secuencia de enteros, $0 < i_1 < i_2 < \cdots < i_k$, tal que se cumple que para la generación g_n, se tiene que: $g_n = g_{n-i_1} g_{n-i_2} \cdots g_{n-i_k}$.

Existen diferentes tipos de L-sistemas, como se mencionó anteriormente, los D0L-sistemas, que son la clase que será usada aquí, hacen alusión a determinista y libre de contexto, propiedades de las reglas de producción definidas para esos sistemas, ya que para esta clase de L-sistemas, cada regla toma en cuenta cada símbolo individualmente, lo que se considera **libre de contexto** y en caso contrario sería condicionar los símbolos vecinos al símbolo o variable de la regla de producción. La propiedad de determinista está presente de igual manera a la hora de definir una regla de producción a cada símbolo, si este tiene más de dos reglas o asignaciones no será determinista, al igual que si no se define regla de producción alguna para algún símbolo usado en las reglas de producción. Los diferentes tipos de L-sistemas nacen precisamente de variar el

g_0	b
g_1	a
g_2	ab
g_3	aba
g_4	$abaaba$
g_5	$abaabaaba$
g_6	$abaabaabaabaaba$
g_7	$abaabaabaabaabaabaabaaba$
g_8	$abaabaabaabaabaabaabaabaabaabaabaabaaba$

Tabla 4.1

Primeras 8 generaciones del Fibonacci L-sistema.

cumplimiento de las propiedades de determinista y libre de contexto, determinando así nuevos tipos de L-sistemas y así, crear nuevas dinámicas entre las reglas de producción que se pueden equiparar con dinámicas de la realidad y así generar aplicaciones sobre estas. A continuación se mostrarán algunos de los ejemplos clásicos de L-sistemas de tipo D0, y que posteriormente se interpretarán gráficamente por medio de lenguaje LOGO o lenguaje de la tortuga al igual que autores como Prusinkiewicz y el mismo Lindemayer.

Ejemplo 4.2.3 (Thue-Morse L-sistema)**.** Sea $G = (\Sigma, \varphi, \omega)$ un L-sistema donde $\Sigma = \{a, b\}$, $\omega = a$ y $\varphi : \Sigma^* \longrightarrow \Sigma^*$ un homomorfismo definido como: $\varphi(a) = ab$ y $\varphi(b) = ba$ y extendida a Σ^* como se mencionó en la Observación 1.4.2. La función de producción que contiene las reglas de producción también puede ser denotado simplemente $\varphi = \{p_1, p_2\}$, con $p_1 = a \to ab$ y $p_2 = b \to ba$. En el Ejemplo 1.4.6 de la sección de homomorfismos uniformes del capítulo

g_0	a
g_1	ab
g_2	$abba$
g_3	$abbabaab$
g_4	$abbabaabbaababba$
g_5	$abbabaabbaababbabaababbaabbabaab$

Tabla 4.2

Primeras 5 generaciones del Thue-Morse L-sistema.

1, se definió el homomorfismo que lleva el mismo nombre que este L-sistema y claramente se puede ver su relación con las reglas de producción definidas para este sistema, sin pérdida de generalidad, son la misma función.

Dos características importantes a resaltar del Thue-Morse L-sistema a la hora de examinar sus generaciones son; se cumple para cada generación g_n su longitud es 2^n, es decir $|g_n| = 2^n$ y además, se puede definir una regla para la generación de g_{n+1} para $n \geq 0$. Esto es, porque g_{n+1} está compuesta de la cadena g_n, concatenada con la cadena denominada en la literatura citada en este documento como palabra espejo de g_n, denotada por $R(g_n)$ y que resulta de remplazar en este caso, los símbolos a por el símbolo b, y los símbolos b por el símbolo a en la cadena g_n. Es decir, $g_{n+1} = g_n R(g_n)$.

Finalmente, este sistema se relaciona directamente con la secuencia Thue-Morse, mostrada en el Ejemplo 3.0.2, porque a es prolongable en φ, se puede considerar así entonces $w := \varphi^w(a)$ y la codificación $\tau : \Sigma \longrightarrow \Sigma_2$, tal que $\tau(a) = 0$ y $\tau(b) = 1$, así se tiene que:

$$\tau(w) = \tau(\varphi^w(a)) = \tau(a)\tau(b)\tau(b)\tau(a)\cdots = 0110100\cdots = (a_n)_{n\geq 0} = u.$$

Basta con una codificación para llegar de una a la otra y viceversa. En el Ejemplo 4.4.1 se muestra una forma de representación gráfica para L-sistema.

Ejemplo 4.2.4. Sea $G = (\Sigma, \varphi, \omega)$ un L-sistema, donde $\Sigma = \{a, b\}$, $\omega = a$ y las reglas de producción para G son:

$$p_1 := a \to ab \text{ y } p_2 := b \to ab.$$

En este L-sistema se puede observar que $|g_n| = 2^n$ al igual que el ejemplo anterior y que $g_{n+2} = (g_1)^{2^{n+1}}$ con $n \geq 0$. Este L-sistema es conocido por generar la curva del Dragón cuando dos formas geométricas son asociadas a los símbolos a y b.

En la Figura 4.2 se muestran formas asociadas a los símbolos a y b, un triángulo rectángulo isósceles tomando como base un vector de coloreado de verde y que dependiendo del símbolo varía la dirección en la que se construye el triángulo.

Las reglas de producción $a \to ab$ y $b \to ab$ son interpretadas gráficamente como la

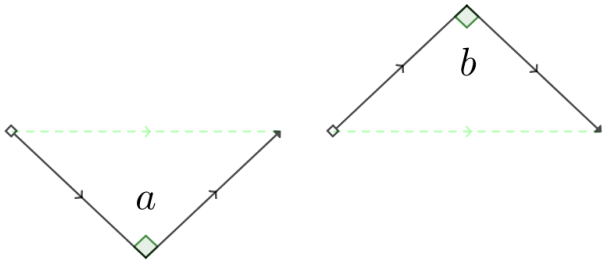


Figura 4.2

Formas asociadas a los símbolos a y b respectivamente en el Ejemplo 4.2.4.

sustitución de los catetos del triángulo rectángulo isósceles que tienen igual longitud por el vector de referencia como base cada uno para dos nuevos triángulos rectángulos isósceles que varían su dirección con respecto a la dirección del vector de referencia generado a partir de la generación inmediatamente anterior como se muestra en la Figura 4.3.

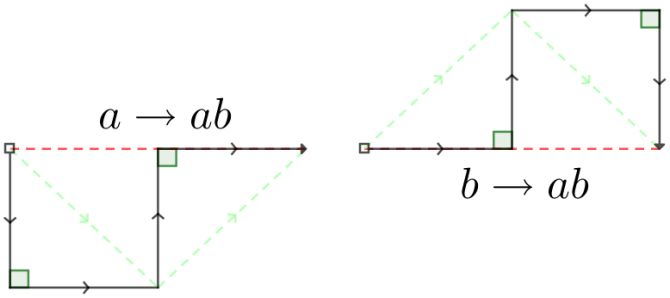


Figura 4.3

Reglas de producción vistas gráficamente a partir de las formas asociadas a los símbolos a y b en el L-sistema del Ejemplo 4.2.4.

Las primeras 6 generaciones de este L-sistema están mostradas en la Tabla 4.3, tanto su cadena, como su representación gráfica creada a partir de la interpretación gráfica que se definió. De igual forma en cada gráfica se muestran los trazos punteados y coloreados de verde que representan a la generación inmediatamente anterior que se usa de referencia para la construcción de esa generación y además el trazo punteado y coloreado de rojo representa la generación anterior a la inmediatamente anterior.

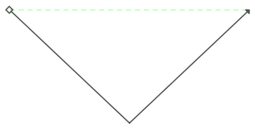
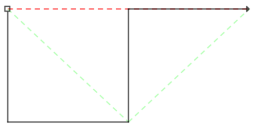
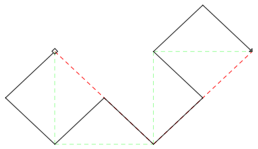
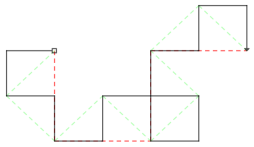
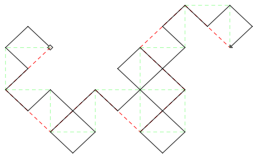
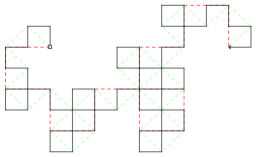
Generación	Cadena	Forma geométrica
g_0	*a*	
g_1	*ab*	
g_2	*abab*	
g_3	*abababab*	
g_4	*ababababababababab*	
g_5	*abababababababababababababababababab*	

Tabla 4.3

Primeras 6 generaciones del L-sistema del Ejemplo 4.2.4.

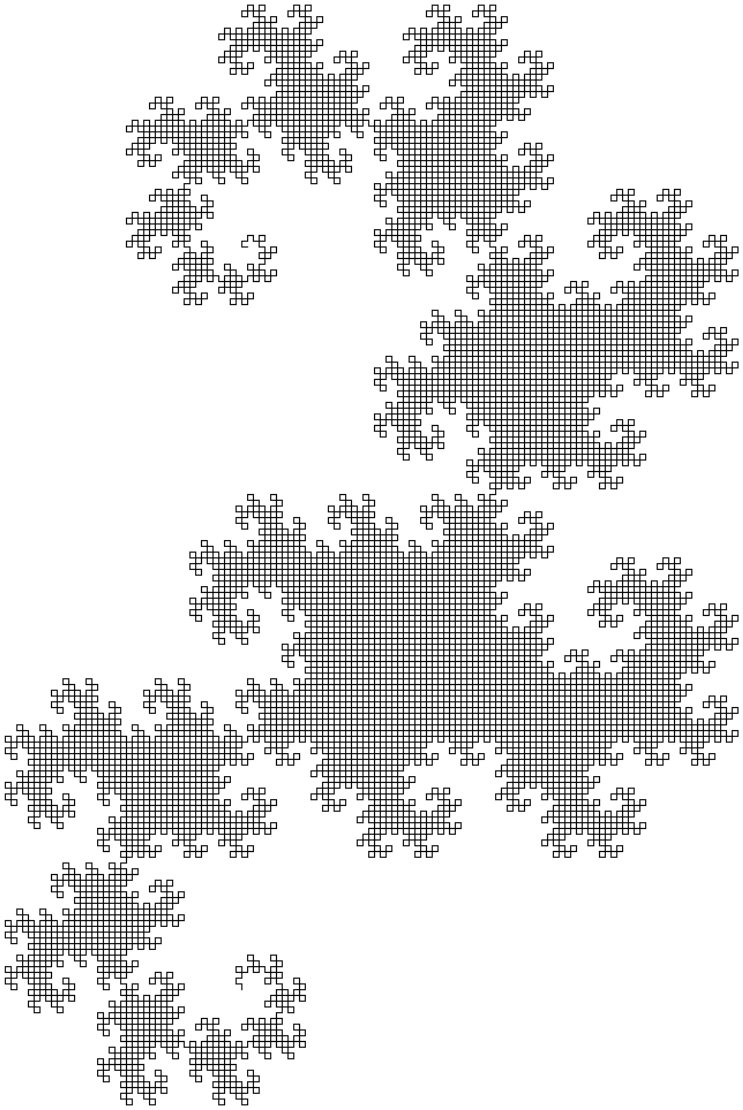


Figura 4.4

Curva del Dragón, Iteración 14, usando método de L-sistema e interpretación geométrica por medio de lenguaje LOGO programado en Python.

Si se continúa iterando y obteniendo más generaciones y considerando esa generación en el límite, se va a encontrar con la curva del Dragón. En la Figura 4.4 se muestra la construcción de la curva del Dragón por medio de un L-sistema diferente que se definirá más adelante y que además representa la misma curva generada por el L-sistema del Ejemplo 4.2.4 pero con una reflexión en su eje por las condiciones iniciales que se programó. La curva del dragón tiene muchas otras propiedades interesantes, entre ellas, una decodificación natural que se desprende de la descripción de los movimientos obtenidos a partir de la dirección de los vectores que componen la curva en cada generación y así se obtiene una nueva secuencia de símbolos en otro alfabeto que curiosamente es análoga a un L-sistema descrito anteriormente.

Antes de repasar esta interesante relación, es necesario definir lo que entendemos como lenguaje de la tortuga o lenguaje LOGO y la forma en como este se relaciona con los L-sistemas y finalmente con las secuencias automáticas y los homomorfismos uniformes.

4.3. Lenguaje de la tortuga

El matemático Seymour Papert es conocido como el inventor de los gráficos de tortugas a partir del lenguaje LOGO, con la publicación de su libro [18]. El lenguaje de la tortuga o lenguaje LOGO es un formalismo o sistema para la traducción de secuencias o cadenas de símbolos a movimientos de un robot cibernético, autómata o como se le denominó finalmente, una "tortuga" mecánica. Curiosamente este sistema fue usado inicialmente como método de programación, con el cual los niños pueden aprender a pensar geométricamente, sin embargo, este también fue ideal a la hora de interpretar geométricamente las dinámicas de los L-sistemas. El matemático Przemysław Prusinkiewicz, que también fue colaborador de Lindenmayer en su artículo [19], describe formalmente la interpretación gráfica de cadenas en base a la geometría de la tortuga, artículo del cuál se invocan algunas definiciones formales que éste usa a la hora de definir esta geometría.

Definición 4.3.1. Una **imagen** Π es un conjunto de puntos en el plano, es decir, $\Pi \subseteq 2^{\mathbb{R}\times\mathbb{R}} = \mathscr{P}(\mathbb{R}\times\mathbb{R})$.

Definición 4.3.2. Una función $I : \Sigma^* \longrightarrow 2^{\mathbb{R}\times\mathbb{R}}$ que mapea o relaciona cadenas sobre Σ con un conjunto de imágenes, es llamada **función de interpretación** (gráfica).

Definición 4.3.3. Un **estado de la tortuga** es una tripleta (x, y, α), donde el par (x, y) representa la posición de la tortuga en coordenadas cartesianas y α representa la dirección en la que apunta la cabeza. Adicionalmente se definen los parámetros d como la longitud de cada **paso** hacia adelante que hace la tortuga y θ como el **ángulo de incremento**. En la Tabla 4.4 se muestra los diferentes comandos y sus símbolos asociados, con una descripción en términos de la posición de la tortuga y los parámetros d, θ, a los que la tortuga puede responder.

Símbolo	Comando
F	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y dibuja su trazo hasta el punto (x', y', α), donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$.
f	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y sin dibujar su trazo hasta el punto (x', y', α), donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$.
$+$	La tortuga en posición (x, y, α) gira a la derecha la cantidad dada por θ finalizando en la posición $(x, y, \alpha + \theta)$ (Se asume que la orientación positiva de los ángulos es en sentido de las manecillas del reloj).
$-$	La tortuga en posición (x, y, α) gira a la izquierda la cantidad dada por θ finalizando en la posición $(x, y, \alpha - \theta)$.
$\vert$	La tortuga se voltea. El cambio de estado de la tortuga, desde un estado inicial (x, y, α) es el nuevo estado, $(x, y, \alpha + 180^\circ)$.

Tabla 4.4

Algunos de los comandos que se pueden definir para la tortuga en el lenguaje LOGO.

La tortuga ignorará los demás símbolos que no denoten un comando previamente establecido a la interpretación gráfica y preserva su estado.

Definición 4.3.4. Sea u una cadena sobre un alfabeto Σ, (x_0, y_0, α_0) el estado inicial de la tortuga y $d,\ \theta$ dos parámetros fijos. La imagen (conjunto de lineas) dibujadas o trazadas por la tortuga respondiendo a la lectura de la cadena u de izquierda a derecha es llamada **interpretación de la tortuga** de u.

Ejemplo 4.3.5. La imagen de la Figura 3.2 es la interpretación de la tortuga para la secuencia que también puede ser generada a partir de la decodificación τ de la secuencia generada por el L-sistema $G = (\Sigma, \varphi, \omega)$, donde $\Sigma = \{A, B, C, D\}$, $\omega = A$ y las reglas de producción están dadas por el homomorfismo 2-uniforme φ definido como:

$$\varphi(A) = AB, \qquad \varphi(B) = AC;$$
$$\varphi(C) = DB, \qquad \varphi(D) = DC.$$

La codificación τ está dada por:

$$\tau(A) = 1, \qquad \tau(B) = 1;$$
$$\tau(C) = -1, \qquad \tau(D) = -1.$$

En el Ejemplo 3.0.3 se mostró que a partir de la regla (3.2) se puede generar una nueva secuencia d_n sobre el alfabeto $\Sigma = \{F, R, L\}$ donde cada símbolo está asociado a los siguientes comandos:

Símbolo	Comando
F	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y dibuja su trazo hasta el punto (x', y', α), donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$.
R	La tortuga gira 90° en el sentido de las masillas del reloj y avanza un paso de longitud dada d en esa misma dirección. El cambio de estado de la tortuga, desde un estado inicial (x, y, α) es el nuevo estado, $(x', y', \alpha - 90^\circ)$, donde $x' = x + d\cos(\alpha + 90^\circ)$ y $y' = y + d\sin(\alpha + 90^\circ)$. Adicionalmente se traza o se dibuja el semento que une los puntos (x, y) y (x', y').
L	La tortuga gira 90° en el sentido contrario de las masillas del reloj y avanza un paso de longitud dada d en esa misma dirección. El cambio de estado de la tortuga, desde un estado inicial (x, y, α) es el nuevo estado, $(x', y', \alpha - 90^\circ)$, donde $x' = x + d\cos(\alpha - 90^\circ)$ y $y' = y + d\sin(\alpha - 90^\circ)$. Adicionalmente se traza o se dibuja el semento que une los puntos (x, y) y (x', y').

Tabla 4.5

Comandos para la tortuga en la generación de la curva 3.2.

Ejemplo 4.3.6. La imagen de la Figura 4.4 es la interpretación de la tortuga para la cadena de la generación 14 para el L-sistema $G = (\Sigma, \varphi, \omega)$, donde $\Sigma = \{A, B, F, T\}$, $\omega = A$ y las reglas de producción están dadas por el homomorfismo 2-uniforme φ definido como:

$$\varphi(A) = AT, \qquad \varphi(B) = BF;$$
$$\varphi(F) = AF, \qquad \varphi(T) = BT.$$

La codificación $\tau = id_{\Sigma}$, en este caso la función identidad, es decir:

$$\tau(A) = A, \qquad \tau(B) = B;$$
$$\tau(F) = F, \qquad \tau(T) = T.$$

Las reglas de dibujo e interpretación de la cadena generada para los símbolos $\{A, B, F, T\}$ está dada por:

Símbolo	Comando
B	La tortuga en posición (x, y, α) avanza un paso a la izquierda de longitud d y dibuja su trazo hasta el punto (x', y, α), donde $x' = x - d$.
F	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y dibuja su trazo hasta el punto (x', y', α), donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$.
A	La tortuga en posición (x, y, α) avanza un paso a la izquierda de longitud d y dibuja su trazo hasta el punto (x', y, α), donde $x' = x + d$.
T	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y dibuja su trazo hasta el punto (x', y', α), donde $x' = x - d\cos(\alpha)$ e $y' = y - d\sin(\alpha)$.

Tabla 4.6

Comandos para la tortuga en la generación de la curva 4.4.

4.4. Curvas, fractales y gráficos generados a partir de secuencias k-automáticas

Como se mostró en el anterior ejemplo, a partir de reglas de dibujo para L-sistemas y a partir de la relación entre homomorfismos k-uniformes y secuencias k-automáticas, es posible generar secuencias infinitas de comandos de dibujo en el lenguaje LOGO que permite obtener diferentes imágenes y curvas, entre ellas la mencionada en el Ejemplo 4.2.4. Esta sección está enfocada en compartir con el lector diferentes comandos y reglas derivadas de L-sistemas ya conocidos que posibilita interpretar gráficamente secuencias automáticas que por su propia definición y equivalencia con los homomorfismos automáticos pueden ser consideradas también

cadenas generadas por L-sistemas.

Ejemplo 4.4.1 (Curva de Koch)**.** En el Ejemplo 3.0.2, se mostró la secuencia Thue-Morse y en Ejemplo 4.2.3 se mostró el L-sistema que puede generarla cuyo homomorfismo empleado es 2-uniforme, es así como esta secuencia es 2-automática, por el teorema de Cobham (3.2.2). Se pueden definir dos comandos en LOGO asociados a los símbolos a y b, y de esta manera generar una curva equivalente a la curva fractal conocida como curva de Koch.[2]

Estos comandos son:

Símbolo	Comando
a	La tortuga en posición (x, y, α) avanza un paso de longitud d, en dirección α y dibuja su trazo hasta el punto (x', y', α), donde $x' = x + d\cos(\alpha)$ e $y' = y + d\sin(\alpha)$.
b	La tortuga en posición (x, y, α) gira 60° en contra de las manecillas del reloj, la nueva posición de la tortuga es $(x, y, (\alpha - 60^\circ))$.

Tabla 4.7

Comandos para la tortuga en la generación de la curva 4.5.

Al aplicar los comandos de la tortuga definidos en la Tabla 4.7 a partir de la aparición del símbolo asociado a cada comando en la secuencia Thue-Morse, $u = (a_n)_{n\geq 0} = abbababbaababbabaab\cdots$, al leer sus términos desde su símbolo inicial a_0 es posible generar una curva equivalente a la curva de Koch, sin embargo en la Figura 4.5, se muestra una parte de la curva generada en la aplicación de estos comandos a la generación 15 de la iteración del homomorfismo que se obtiene la secuencia de Thue-Morse bajo a, es decir, a los primeros $|g_{15}| = 2^{15} = 32768$ términos.

[2]La curva de Koch es un curva fractal muy conocido inicialmente descrita por el matemático sueco Helge von Koch en 1904.

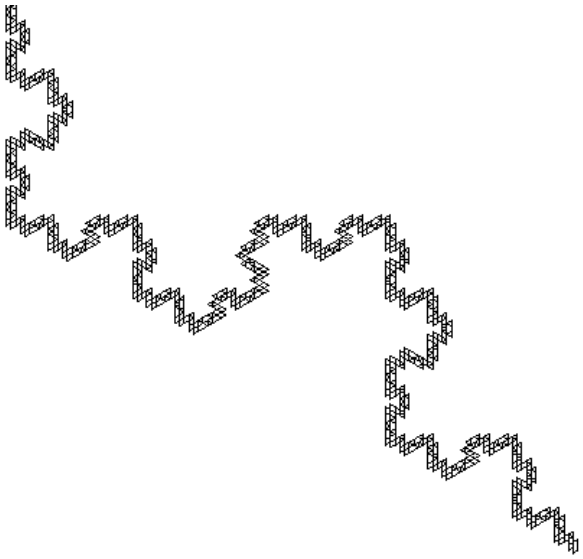


Figura 4.5

Curva Koch, Iteración 15, usando método de L-sistema e interpretación geométrica por medio de lenguaje LOGO programado en Python.

Con algunas modificaciones a los comandos anteriores se obtienen algunas variaciones de la curva 4.5, de las cuales se muestran las siguientes tres:

(1) La primera variación usa los comandos de la Tabla 4.8, los cuales se dan en términos de comandos anteriormente mostrados en la Tabla 4.4 para las constantes $d = 5, \theta = 60^\circ$ y $\alpha_0 = 90^\circ$. La tabla que muestra la codificación de los símbolos a y b en comandos anteriormente definidos es:

Símbolo	Comando
a	$F-$
b	$---$

Tabla 4.8

Comandos de la primera variación de la curva 4.5.

Al aplicar los comandos de la Tabla 4.8 a la lectura de la secuencia Thue-Morse, se obtiene la curva:

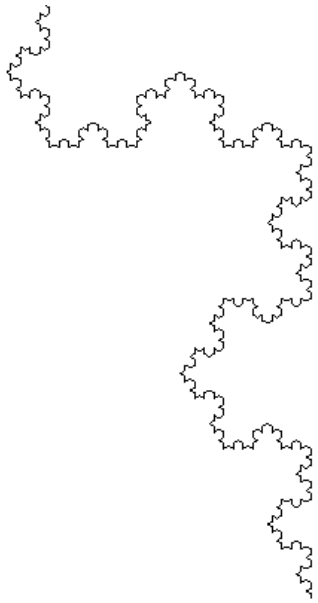


Figura 4.6

Primera variación de la curva 4.5

(2) Los comandos de la Tabla 4.9, los cuales se dan en términos de comandos anteriormente mostrados en la Tabla 4.4, para las constantes $d = 5,\ \theta = 180^\circ,\ \theta' = 15^\circ$, y $\alpha_0 = 60^\circ$. La tabla que muestra la codificación de los símbolos a y b en comandos anteriormente definidos es:

Símbolo	Comando
a	$-$
b	$-_{\theta'}F$

Tabla 4.9

Comandos de la segunda variación de la curva 4.5.

Donde el comando $-_{\theta'}$ es igual que $-$, pero en base al ángulo θ'. Al aplicar los comandos de la Tabla 4.9 a la lectura de la secuencia Thue-Morse, se obtiene la curva:

(3) Finalmente, en la tercera variación, los comandos de la Tabla 4.10, los cuales se dan en términos de comandos anteriormente mostrados en la Tabla 4.4 para las constantes $d = 5,\ \theta = 180^\circ,\ \theta' = 120^\circ$, y $\alpha_0 = 145^\circ$. La tabla que muestra la codificación de los

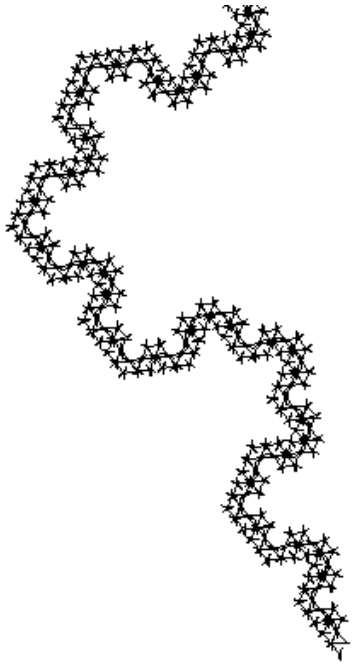


Figura 4.7

Segunda variación de la curva 4.5

símbolos a y b en comandos anteriormente definidos es:

Símbolo	Comando
a	$-F$
b	$-_{\theta'}F$

Tabla 4.10

Comandos de la tercera variación de la curva 4.5.

Donde el comando $-_{\theta'}$ es igual que $-$, pero en base al ángulo θ'. Al aplicar los comandos de la Tabla 4.9 a la lectura de la secuencia Thue-Morse, se obtiene la curva:

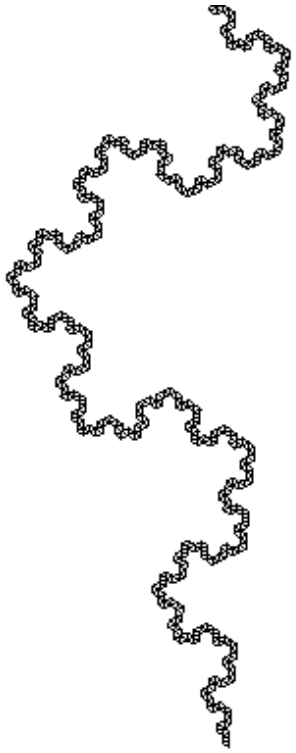


Figura 4.8

Tercera variación de la curva 4.5

Ejemplo 4.4.2 (Curva de Lévy C)**.** Al considerar curva generada a partir del L-sistema $G = (\Sigma' = \{a,\ b,\ c,\ d\},\ \varphi,\ \omega = a)$, donde las reglas de producción para G son $p_1 = a \to bc,\ p_2 = b \to cd,\ p_3 = c \to da$ y $p_4 = d \to ab$, con φ un homomorfismo 2-uniforme, por el teorema de Cobham (3.2.2) es posible crear un 2-AFDS $\mathbb{M} = \{Q,\ \Sigma_2,\ \delta,\ q_0,\ \Delta,\ \tau\}$, con $Q = \Sigma' = \Delta,\ \Sigma = \Sigma_2,\ q_0 = a,\ \tau = id_Q$ y la función de transición $\delta : Q \times \Sigma \longrightarrow Q$, definida para todo $q \in Q$ y $a \in \Sigma$ como: $\delta(q,a) = \varphi(q)[a]$. Es así que por la construcción semejante a la empleada en la demostración del Teorema 3.2.2, entonces se cumple que $\varphi^w(a) = (\tau(\delta(q_0,(n)_k)))_{n\geq 0}$. En la Figura 3.3 del Ejemplo 3.2.5 se muestra el diagrama de transición para $\mathbb{M}$. Aplicando la codificación τ' a la secuencia $\varphi^w(a)$, donde: $\tau'(a) = B,\ \tau'(b) = F,\ \tau'(c) = A$ y $\tau'(d) = T$ y así, la secuencia $u = \tau'(\varphi^w(a))$ con los comandos asociados a los símbolos en $\{A,\ B,\ F,\ T\}$ iguales a los definidos en la Tabla 4.6 es posible generar la curva de la Figura 4.9 conocida como la curva de Lévy.[3]

[3]La curva de Lévy C es un fractal autosimilar que lleva el nombre del matemático francés Paul Pierre Lévy quien, en 1938, fue el primero en exhibir sus propiedades de autosimilitud y proveer una construcción geométrica.

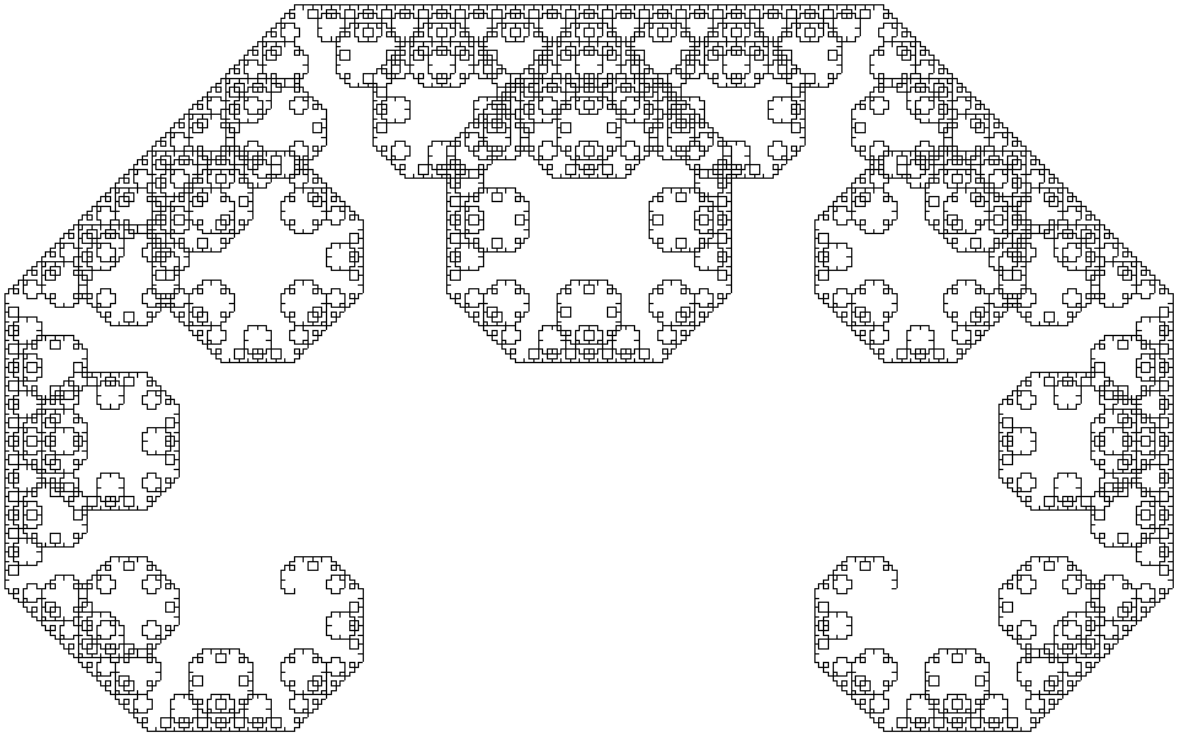


Figura 4.9

Curva de Lévy C construida en el Ejemplo 4.4.2

Ejemplo 4.4.3 (Curva de Hilbert)**.** Sea $\Sigma = \Sigma_8 = \{0,\ 1, \ldots,\ 7\}$ y $\Delta = \{A, B, F, T\}$ dos alfabetos. Considerando el homomorfismo 4-uniforme $\varphi : \Sigma^* \longrightarrow \Sigma^*$, formado por las imágenes de las entradas:

$$\begin{array}{lll} 0 \longrightarrow 0123, & 1 \longrightarrow 1045, & 2 \longrightarrow 1046; \\ 3 \longrightarrow 7650, & 4 \longrightarrow 0127, & 5 \longrightarrow 6731; \\ 6 \longrightarrow 6732, & 7 \longrightarrow 7654. & \end{array}$$

El homomorfismo φ es prolongable en el símbolo 0 y así, la secuencia $u = \varphi^w(0)$ es 4-automática.

Si se considera la codificación $\tau : \Sigma^* \longrightarrow \Delta^*$, formado por imágenes de las entradas:

$$\begin{array}{lll} 0 \longrightarrow F, & 1 \longrightarrow A, & 2 \longrightarrow T; \\ 3 \longrightarrow A, & 4 \longrightarrow B, & 5 \longrightarrow F; \\ 6 \longrightarrow B, & 7 \longrightarrow T. & \end{array}$$

Así, que la secuencia $v = \tau(\varphi^n(0))$ es también 4-automática y al usar los mismos comandos asociados a los símbolos en la Tabla 4.6 a la lectura de la secuencia v, es posible

generar la denominada curva de Hilbert.[4] En la Figura 4.10 se muestra la curva resultante para la cadena formada en la quinta iteración, es decir $\tau(\varphi^5(0))$ y en la Figura 4.11 se muestra la curva asociada para $n = 7$.

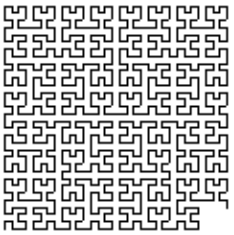


Figura 4.10

Curva de Hilbert, generada a partir de la cadena $\tau(\varphi^5(0))$.

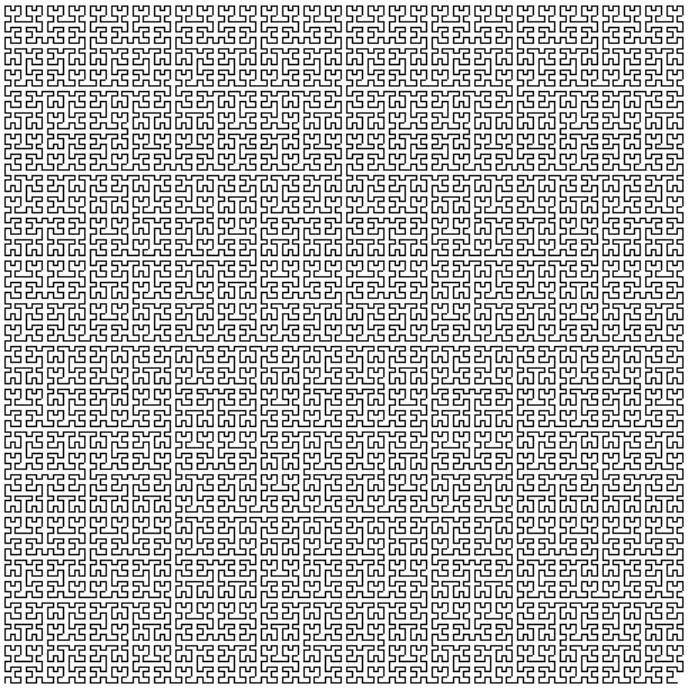


Figura 4.11

Curva de Hilbert, generada a partir de la cadena $\tau(\varphi^7(0))$.

Otra manera de interpretaciones gráficas de las secuencias *k*-automáticas, es por medio de la coloración de píxeles (cuadrados con áreas constantes) con un color asociado a los símbolos sobre los cuales la secuencias se encuentran definidas. Por ejemplo, para la secuencia

[4]La curva de Hilbert (también conocida como la curva que recubre el plano de Hilbert) es una curva fractal continua que recubre el plano descrita inicialmente por el matemático alemán David Hilbert en 1891.

Thue-Morse, $u = abbabaabbaababb\cdots$, que puede ser definida como $\varphi^w(a)$, con φ igual al dado en el Ejemplo 4.2.3, se puede asignar o asociar los colores negro al símbolo a y blanco al símbolo b y al dibujar de forma lineal los píxeles con el color correspondiente a cada símbolo de la secuencia Thue-Morse, es posible generar la Figura 4.12.

Figura 4.12

Primeros 16 *términos de la secuencia Thue-Morse en una sola fila.*

La figura anterior podría volver a dibujarse como una concatenación vertical de filas de igual longitud de píxeles y así formar una imagen de 2 dimensiones a diferencia de la Figura 4.12 como se puede ver en la Figura 4.13.

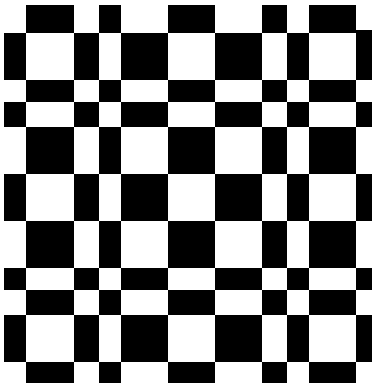


Figura 4.13

Primeros 256 *términos de la secuencia Thue-Morse en* 16 *filas dibujadas una sobre la otra sucesivamente.*

Esta clase de gráficos tienen sus análogos y pueden ser generados por secuencias automáticas denominadas multidimensionales. Por ejemplo, las secuencias automáticas bidimensionales también llamadas tablas o matrices. Para ello hay que volver al inicio, cuando se definió el concepto de cadena, este primer concepto se definió a partir de un conjunto finito de símbolos denominado alfabeto, ahora bien, se pueden considerar dos alfabetos finitos Σ, Δ

y considerar su producto cartesiano, es decir, $\Sigma \times \Delta = \{(a,b) : \ a \in \Sigma \text{ y } b \in \Delta\}$ como un nuevo alfabeto. Las cadenas formadas sobre este nuevo alfabeto forman simultáneamente dos cadenas de igual longitud sobre los alfabetos Σ y Δ respectivamente. Por ejemplo, si $\Sigma = \{a,m,o,r\}$ y $\Delta = \{a,b,s\}$, entonces las cadenas sobre $\Sigma \times \Delta$ tienen dos posibles notaciones, por ejemplo:

$$(a,c)(m,a)(o,s)(r,a) = amor \times casa.$$

Antes de continuar, hay que ver la siguiente figura que puede ser generada por una secuencia automática bidimensional.

Ejemplo 4.4.4 (Alfombra de Sierpinski)**.** Considerando la matriz $\mathbf{s} = (S_{i,j})_{i,j\geq 0}$ sobre $\Sigma_2 = \{0,1\}$, donde cada $S_{i,j}$ es definido como:

$$S_{i,j} = \begin{cases} 0, \text{ si existe al menos un } k \geq 0 \text{ entero } : (i)_3[k] = (j)_3[k]; \\ 1, \text{ en caso contrario.} \end{cases}$$

Las primeras 9 filas y columnas de esta secuencia están dadas por la matriz:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.3}$$

A partir de las cadenas definidas sobre el producto cruz de dos alfabetos es posible construir autómatas con entradas dobles que reconozcan palabras en paralelo y no en serie, por ejemplo, considerando $\Sigma = \Delta = \Sigma_2 = \{0,1\}$, entonces el autómata definido sobre el alfabeto $\Sigma \times \Delta$ dado por el diagrama de transición mostrado en la Figura 4.14 corresponde precisamente al valor S_i, j, para $i, j \geq 0$ dos enteros.

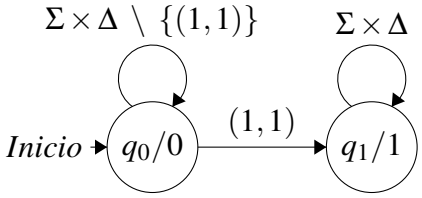


Figura 4.14

A

Es posible generalizar la teoría de secuencias automáticas consideradas uno-dimensionales, las cuales fueron las que se definieron a lo largo de los capítulos anteriores, a matrices automáticas d-dimensionales. Como se mencionó anteriormente, para el caso $d = 2$, una tabla infinita k-automática, puede ser computada por un autómata que toma los índices de la fila y la columna como entradas, en paralelo y expresados en base k, como se vio para el autómata de la Figura 4.14 que precisamente es el que generadora de la matriz de la alfombra de Sierpinski, que genera la Figura 4.15.

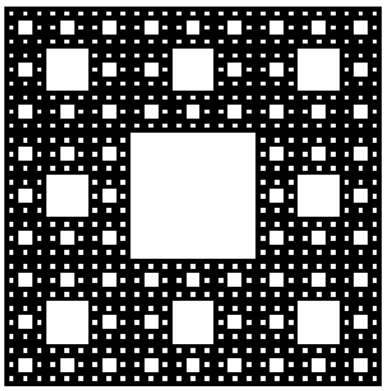


Figura 4.15

Alfombra de Sierpinski, representación por píxeles de $\varphi^4(1)$.

La matriz **s** también puede ser generada por la iteración de un homomorfismo, en este caso un homomorfismo bidimensional, lo cual significa que cada la imagen de cada símbolo bajo el homomorfismo es una matriz cuadrada de símbolos. Es decir, definiendo el homomorfismo φ inicialmente para los símbolos en $\Sigma_2 = \{0,1\}$, como:

$$1 \to \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad 0 \to \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

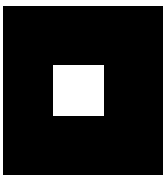


Figura 4.16

Representación por píxeles de $\varphi(1)$.

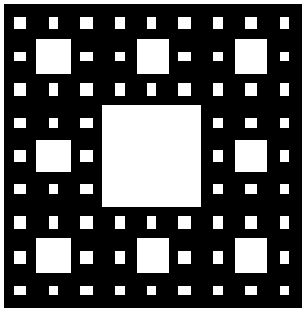


Figura 4.17

Representación por píxeles de $\varphi^3(1)$.

Luego de extender φ sobre las matrices sobre Σ_2, es posible obtener $\mathbf{s}$, al iterar el homomorfismo φ sobre el punto fijo 1, es decir $\mathbf{s} = \varphi^w(1)$. Note que $A = \varphi^2(1)$ en 4.3.

4.5. Conclusiones

A partir de los diferentes conceptos, definiciones y resultados mostrados a lo largo del desarrollo de este trabajo se pudo describir a las secuencias k-automáticas de una manera robusta desde sus preliminares en los campos de la combinatoria de palabras y autómatas finitos, y así se pudo demostrar la equivalencia con los homomorfismos de cadenas k-uniformes bajo ciertas condiciones como se demostró anteriormente; siendo este el objetivo principal de este documento. A partir de esta relación se logra evidenciar también una estrecha relación entre la propiedad de autosimilaridad propia de las curvas fractales y la propiedad de periodicidad de una secuencia k-automática, como se mostró en algunos de los ejemplos en el último capítulo.
De este mismo modo, se pudo demostrar que algunas curvas fractales también pueden ser generadas o construidas a partir de la aplicación de ciertos comandos y reglas de dibujo definidos para L-sistemas y propios del lenguaje LOGO sobre una codificación de alguna secuencia k-automática.
Para dibujar o graficar las secuencias automáticas, fue necesario crear un algoritmo en el lenguaje de programación Python que a partir del resultado principal de este documento que caracteriza estas secuencias como el punto iterativo para un homomorfismo k-uniforme, es posible iterar hasta una cierta cantidad de símbolos y de esta manera usando paquetes previamente programados sobre el lenguaje LOGO es posible reproducir esos comandos en una ventana de dibujo virtual, la cual va reproduciendo la lista de comandos dados por la secuencia y de esta forma obtener las figuras que aquí se mostró.
Finalmente, se resalta el carácter introductorio de este documento en esta área de la matemática, que sirve como material de estudio y aprendizaje del modelo matemático computacional denominado autómata finito y que plantea la resolución y una descripción del procesamiento, lectura y escritura de lenguajes a partir de estos, donde se pudo evidenciar una gran cantidad de estudios relacionados con estos objetos matemáticos, sin embargo, en su mayoría en el idioma inglés, extendiendo así la motivación de poder seguir desarrollando, estudiando y divulgando estos tópicos matemáticos al resto de la comunidad académica y en general.

A. Anexos

A.1. Método para generar imágenes

Se utilizaron dos diferentes técnicas a la hora de generar las imágenes que aquí se presentaron; la primera técnica que fue utilizada para realizar las interpretaciones geométricas de cada generación en la tabla 4.3 fue por construcción en el programa Goegebra 5.0[1], conservando las propiedades, formas y reglas de dibujo asociadas a los símbolos *a* y *b* para cada generación mostrada en la tabla.

En la figura A.1 del anexo A.2 se puede observar la cantidad de elementos usados para garantizar que la construcción cumpla con las propiedades con las cuales se definieron las formas asociadas a *a* y *b* en el ejemplo 4.2.4.

La segunda técnica utilizada para las demás figuras mostradas es por medio de la programación de diferentes algoritmos en el lenguaje Python que permiten emular las diferentes configuraciones matemáticas, como los comandos asociados con el lenguaje de la tortuga, los colores asociados a los comandos, la toma de imágenes y entre otros como se mostró a lo largo del texto. En particular, el algoritmo presentado en [20], fue utilizado como base para la programación del mismo, así como se resalta también el uso de los diferentes paquetes que ya existen en este lenguaje de programación, como lo es `turtle`,`tkinter` y `numby`, que facilita el implementación de los comandos que hacen mover a la tortuga. En el siguiente [repositorio] se encuentra el código fuente en lenguaje python usado en este documento.

[1] Geogebra es un software libre de visualización de dinámicas matemáticas, creado por Markus Hohenwarter en 2001 con el fin de ser utilizado en el aprendizaje de las matemáticas.

A.2. Captura de pantalla de la construcción realizada para la generación de las imágenes usadas en la tabla 4.3.

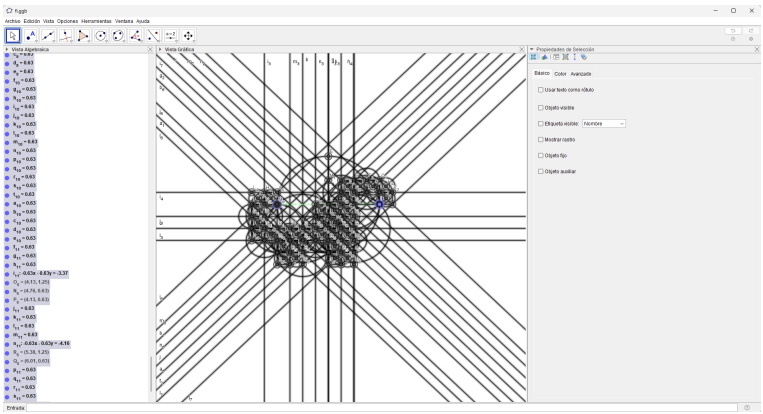


Figura A.1

Captura de pantalla de la construcción realizada para la generación de las imágenes usadas en la tabla 4.3.

Bibliografía y Cibergrafía

[1] Alan Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical systems theory*, 3(2):186–192, 1969.

[2] Alan Cobham. Uniform tag sequences. *Mathematical systems theory*, 6(1):164–192, 1972.

[3] Jean-Paul Allouche, Jeffrey Shallit, et al. *Automatic sequences: theory, applications, generalizations*. Cambridge university press, 2003.

[4] Jeffrey Shallit. *The logical approach to automatic sequences: Exploring combinatorics on Words with Walnut*, volume 482. Cambridge University Press, 2022.

[5] José L Ramírez and Gustavo N Rubiano. Generación de curvas fractales a partir de homomorfismos entre lenguajes [con mathematica®]. *Revista Integración*, 30(2):129–150, 2012.

[6] JM Campbell. Artwork based on automatic sequences. *Journal of Mathematics and the Arts*, 16(4):287–308, 2022.

[7] Hans Petter Langtangen. *Python scripting for computational science*. Springer, 2008.

[8] Thomas A Sudkamp. *Languages And Machines: An Introduction To The Theory Of Computer Science, 3/E*. Pearson Education India, 2007.

[9] Rodrigo de Castro Korgi. Teoría de la computación: lenguajes, autómatas, gramáticas. *Editorial UN*, 2004.

[10] Jean-Paul Allouche and Jeffrey Shallit. Automatic sequences are also non-uniformly morphic. In *Discrete Mathematics and Applications*, pages 1–6. Springer, 2020.

[11] Jibitesh Mishra and Sarojananda Mishra. *L-system Fractals*. Elsevier, 2007.

[12] Gary William Flake. *The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation*. MIT press, 2000.

[13] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[14] Karel Culik and Simant Dube. L-systems and mutually recursive function systems. *Acta Informatica*, 30:279–302, 1993.

[15] Manuel Alfonseca and Alfonso Ortega. A study of the representation of fractal curves by l systems and their equivalences. *IBM Journal of Research and Development*, 41(6):727–736, 1997.

[16] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

[17] Lindenmayer Aristid. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, 1968.

[18] Seymour A Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic books, 2020.

[19] Przemyslaw Prusinkiewicz. Graphical applications of l-systems. In *Proceedings of graphics interface*, volume 86, pages 247–253, 1986.

[20] Gianni Perez. L-system-drawing (algoritmo en python). `https://github.com/ambron60/l-system-drawing`, 2021. Accedido: 2023–08-25.