Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Clausura Conmutativa de Lenguajes Regulares

Tesis de Licenciatura en Ciencias de la Computación

Simón Lew Deveali

Directora: Dra. Verónica Becher
Codirector: Lic. Ignacio Mollo
Buenos Aires, 2024

CLAUSURA CONMUTATIVA DE LENGUAJES REGULARES

Consideremos un alfabeto finito A y A^* el monoide libre generado por el alfabeto con la operación de concatenación. Dos palabras comparten su imagen conmutativa cuando una es permutación de los símbolos de la otra palabra. La clausura conmutativa de un lenguaje $L \subseteq A^*$ es el conjunto $\mathcal{C}(L) \subseteq A^*$ de palabras cuya imagen conmutativa coincide con la de alguna palabra en L. Damos un algoritmo que, dado un lenguaje regular L, produce el autómata finito determinístico que acepta la clausura conmutativa $\mathcal{C}(L)$ siempre que ésta sea regular. El problema de decidir si $\mathcal{C}(L)$ es regular ya fue resuelto por Ginsburg y Spanier en 1966 utilizando la decibilidad de las sentencias de Presburger, y por Gohon en 1985 proponiendo un algoritmo mas simple basado en series formales. Sin embargo, hasta la fecha la literatura no tiene un algoritmo que en todos los casos positivos construya un autómata o una expresión regular que acepte $\mathcal{C}(L)$. Este es el principal aporte de nuestro trabajo.

Palabras claves: lenguajes regulares, clausura conmutativa, autómatas, monoide libre conmutativo, conjuntos reconocibles.

COMMUTATIVE CLOSURE OF REGULAR LANGUAGES

Let us consider a finite alphabet A and A^*, the free monoid generated by the alphabet with the concatenation operation. Two words share their commutative image when one is a permutation of the symbols of the other word. The commutative closure of a language $L \subseteq A^*$ is the set $\mathcal{C}(L) \subseteq A^*$ of words whose commutative image coincides with that of some word in L. We provide an algorithm that, given a regular language L, produces the deterministic finite automaton that accepts the commutative closure $\mathcal{C}(L)$, provided that this closure is regular. The problem of deciding whether $\mathcal{C}(L)$ is regular was already solved by Ginsburg and Spanier in 1966 using the decidability of Presburger sentences, and by Gohon in 1985 with a simpler algorithm based on formal power series. However, to date, the literature does not contain an algorithm that, in all positive cases, constructs an automaton or a regular expression that accepts $\mathcal{C}(L)$. This is the main contribution of our work.

Keywords: regular languages, commutative closure, automata, free commutative monoid, recognisable sets.

Índice general

1. ¿CUÁL ES EL PROBLEMA?

Definiciones generales

Un alfabeto es un conjunto finito, no vacío, de símbolos. Lo denotamos A. A los elementos de A los llamamos letras, y las secuencias finitas de letras las llamamos palabras. Denotamos A^* al conjunto de las todas las palabras sobre el alfabeto A.

Un lenguaje sobre A, o lenguaje en A^*, es cualquier conjunto de palabras escritas con letras de A. También lo podemos definir como un subconjunto de A^*, es decir, un elemento de $\mathcal{P}(A^*)$.

Definición 1.1 (Autómata Finito)**.** *Un autómata finito $\mathcal{A}= \langle Q, A, E, I, T\rangle$ se especifica con los siguientes elementos:*

- *un conjunto no vacío Q, llamado el conjunto de estados de $\mathcal{A}$*
- *un conjunto A, también no vacío, llamado el alfabeto de $\mathcal{A}$*
- *dos subconjuntos I y T de Q, donde I es el conjunto de estados iniciales y T es el conjunto de estados finales de $\mathcal{A}$.*
- *un conjunto E de $Q \times A \times Q$, llamado el conjunto de transiciones de $\mathcal{A}$.*

Si $e = (p, a, q)$ es una transición del autómata $\mathcal{A}$, es decir, $e \in E$, decimos que a es la etiqueta de e y escribimos $p \xrightarrow{a} q$, o $p \xrightarrow[\mathcal{A}]{a} q$ si no esta claro el autómata al que nos referimos. También decimos que p es el origen y q el destino de la transición e.

Definición 1.2 (Autómata Finito Determinístico (AFD))**.** *Un autómata finito $\mathcal{A} = \langle Q, A, E, I, T\rangle$ es determinístico si y sólo si:*

- *hay exactamente un estado inicial, $|I| = 1$.*
- *para todo $p \in Q$ y para todo $a \in A$ existe a lo sumo una transición en E con origen p y etiqueta a.*

Definición 1.3. *Un cómputo c en $\mathcal{A}$ es una secuencia de transiciones donde el origen de cada una es el destino de la anterior, entonces podemos escribir:*

$$c = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$$

El estado p_0 es el origen del cómputo c, y p_n su destino. La longitud del cómputo c es n, la cantidad de transiciones que componen a c. La etiqueta de c es la concatenación (producto) de las etiquetas de cada transición de c; en el caso de arriba, la etiqueta que se corresponde con c es $a_1a_2 \cdots a_n$ por eso escribimos:

$$c = p_0 \xrightarrow{a_1a_2\cdots a_n} p_n \text{ o } p_0 \xrightarrow[\mathcal{A}]{a_1a_2\cdots a_n} p_n$$

Un cómputo en el autómata $\mathcal{A}$ es exitoso si su origen es un estado inicial y su destino es un estado final. Decimos que una palabra en A^* es aceptada por $\mathcal{A}$ si es la etiqueta de un cómputo exitoso en $\mathcal{A}$.

Los cómputos de longitud 1 en $\mathcal{A}$ son exactamente sus transiciones. Por convención, consideramos que cada estado es origen y destino de un (único) cómputo de longitud 0 cuya etiqueta es la palabra vacía λ, la única palabra en A^* de longitud 0. Siguiendo esta convención: La palabra vacía λ es aceptada por un autómata finito $\mathcal{A}$ si y sólo si hay un estado de $\mathcal{A}$ que es inicial y final ($I \cap T \neq \emptyset$).

Definición 1.4. *El lenguaje aceptado por el autómata finito $\mathcal{A}$, lo notamos $L(\mathcal{A})$ o $|\mathcal{A}|$, es el conjunto de palabras reconocidas por $\mathcal{A}$:*

$$L(\mathcal{A}) = \{w \in A^* \mid \exists p \in I, \exists q \in T \ p \xrightarrow[\mathcal{A}]{w} q\}.$$

Definición 1.5 (Expresión regular)**.** *Una expresión regular sobre el alfabeto A es una formula obtenida inductivamente a partir de los elementos de A y las funciones: $\{\emptyset, \cup, \lambda, \cdot, *\}$ de la siguiente manera:*

I. *$\emptyset$, λ y cualquier letra de A, son expresiones regulares.*

II. *si E y F son expresiones regulares, luego $(E \cup F)$, $(E \cdot F)$ y (E^*) también son expresiones regulares.*

Definición 1.6 (Lenguaje denotado por una expresión regular)**.** *El lenguaje denotado por E, al que nos referimos como*
$L(E)$ o $|E|$, está definido inductivamente en la profundidad de la expresión:

1. *Para las expresiones atómicas:*
   $L(\emptyset) = \emptyset$, $L(\lambda) = \{\lambda\}$ y $L(a) = \{a\}$ para cualquier letra $a \in A$.

2. *Para expresiones compuestas:*
   $L(E \cup F) = L(E) \cup L(F)$, $L(EF) = L(E)L(F)$ y $L(E^*) = L(E)^*$.

Decimos que E denota el lenguaje $L(E)$.

Definición 1.7 (Lenguaje Regular)**.** *Un lenguaje es regular si existe un autómata finito (determinístico) que lo acepta. Equivalentemente, si existe una expresión regular que lo denota.*

Clausura conmutativa

Notación. *Escribimos $|w|_a$ para denotar la cantidad de ocurrencias del símbolo a en la palabra w. Por ejemplo $|010|_0 = 2$.*

Para definir la clausura conmutativa necesitamos el concepto de permutación.

Definición 1.8 (Permutación)**.** *Dos palabras w y $v \in A^*$ son permutación si solo si para toda letra a del alfabeto tenemos exactamente la misma cantidad de apariciones de a en v y w. Es decir, v y w son permutación si para todo $a \in A$, $|w|_a = |v|_a$.*

Podemos definir una función que dada una palabra nos devuelve la cantidad de apariciones de cada letra del alfabeto en esa palabra:

Definición 1.9 (Imagen conmutativa)**.** *Dado un alfabeto A de k símbolos, $A = \{a_1, a_2, ..., a_k\}$, la imagen conmutativa $\varphi : A^* \to \mathbb{N}^k$ para una palabra $w \in A^*$ se define como:*

$$\varphi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$$

La imagen conmutativa de un lenguaje $L \subseteq A^*$ es $\varphi(L) \subseteq \mathbb{N}^k$,

$$\varphi(L) = \{\varphi(w) : w \in L\}.$$

Comentario. *La imagen conmutativa también se conoce como el morfismo de Parikh.*

Luego podemos definir la clausura conmutativa de un lenguaje como:

Definición 1.10 (Clausura Conmutativa)**.** *Sea A un alfabeto de k símbolos y sea φ la imagen conmutativa de las palabras sobre A. Si $L \subseteq A^*$, la clausura conmutativa de L, $\mathcal{C}(L) \subseteq A^*$,*

$$\begin{aligned}\mathcal{C}(L) &= \{w \in A^* \mid \varphi(w) \in \varphi(L)\} \\ &= \varphi^{-1}(\varphi(L)).\end{aligned}$$

Comentario. *Notemos que la clausura conmutativa de un lenguaje regular no es siempre regular. Por ejemplo $\mathcal{C}((ab)^*) = \{w \in A^* \ : \ |w|_a = |w|_b\}$.*

Comentario. *En otros trabajo se estudian las clases de lenguajes cerradas bajo conmutabilidad. En particular en [1] definen la clase robusta $\mathcal{W}$ cerrada por la conmutación. Esta, puede ser definida como la variedad positiva mas grande de lenguajes que no contienen $(ab)^*$.*

Problema. *Dada una expresión regular en A^* dar una autómata finito para su clausura conmutativa en el caso que esta sea regular.*

Saber si la clausura conmutativa de un lenguaje es regular ya fue resuelto exitosamente por Ginsburg y Spanier en 1966, y por Gohon en 1985. La primera prueba es indirecta y utiliza la decibilidad de las sentencias de Presburger. La segunda da un algoritmo más simple basándose en series formales y el resultado de Eilenberg y Schutzenberger [3]. El algoritmo de Gohon lo describimos en el siguiente capítulo.

Proposición 1 ([4, 5])**.** *Es decidible determinar si la clausura conmutativa de un lenguaje regular en A^* es regular.*

Si bien el problema de decisión ya se encuentra resuelto, nuestro objetivo es dar un algoritmo que construya el autómata. Probaremos dando un algoritmo (Capítulo 5) el siguiente teorema:

Teorema 1. *Dada una expresión regular de un lenguaje L en A^* cuya clausura conmutativa $\mathcal{C}(L)$ es regular, podemos construir de forma efectiva un autómata para la clausura conmutativa.*

Ejemplo 1.1 (Clausura conmutativa regular)**.** *La clausura conmutativa del lenguaje denotado por $b(a^2 \cup b^2)^*$ es regular. El lenguaje consiste de las palabras con una cantidad par de a e impar de b. Nuestro algoritmo produce el siguiente autómata (Figura 1.1)*

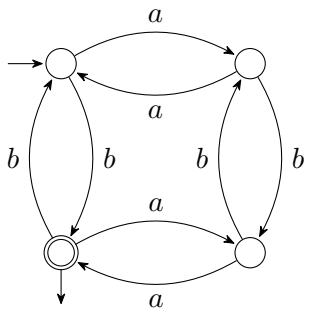

Figura 1.1: Autómata de $\mathcal{C}(L)$ con $L = |b(a^2 \cup b^2)^*|$.

Para algunas familias de lenguajes regulares el problema de construir el autómata finito que acepte la clausura conmutativa de un lenguaje regular ya fue estudiado. En [1] se prueba que la clausura conmutativa de lenguajes de grupo es regular y Hoffmann en [6] no solamente da una construcción sino que además da una cota de la cantidad de estados del autómata resultante.

Un lenguaje es de grupo si existe un autómata de permutación que lo acepte.

Definición 1.11 (Autómata de permutación [11])**.** *Un AFD completo es de permutación si y sólo si para todo $q \in Q$, $a \in A$ existe una sola transición en E con destino q y etiqueta a.*

Comentario. *Los autómatas de la Figura 1.1 y Figura 1.2 son de permutación.*

Ejemplo 1.2. *Autómata de permutación cuyo lenguaje aceptado L no es conmutativo, es decir $\mathcal{C}(L) \neq L$.*

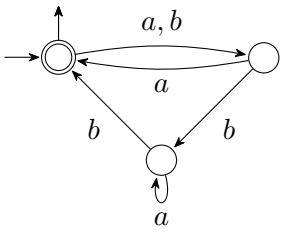

Figura 1.2: Autómata de $|\big((a \cup b)(a \cup ba^*b)\big)^*|$.

2. EL ALGORITMO DE DECISIÓN DE GOHON

El algoritmo de Gohon [5] decide si la clausura conmutativa de un lenguaje en A^* es regular. Pero en lugar de partir del lenguaje parte de la imagen conmutativa del mismo. Es decir, este es un algoritmo de decisión directamente en $\mathbb{N}^k$, el monoide libre conmutativo.

Definición 2.1 (Monoide)**.** *Un monoide es un conjunto M equipado con una operación binaria asociativa y un elemento neutro para esa operación. A la operación se la llama producto. Al elemento neutro lo llamaremos λ.*

La operación sobre M se escribe usualmente con un punto, es decir escribimos $m \cdot n$. También se puede obviar el símbolo y se puede escribir directamente la concatenación de los operandos, es decir escribimos mn. Cuando la operación sobre el monoide M además es conmutativa, es decir, para todo m y n en M vale que $mn = nm$, se la suele notar con $+$ y se suele notar el elemento neutro con 0. Cuando la operación es conmutativa decimos que M es conmutativo.

Definición 2.2 (Monoide Libre)**.** *Un monoide libre generado por un conjunto A, es el monoide de todas las palabras sobre el alfabeto A, con la concatenación de las palabras como la operación. Y el elemento neutro es la palabra vacía.*

Definición 2.3 (Monoide Libre Conmutativo $\mathbb{N}^k$)**.** *Sea A un alfabeto. El monoide libre conmutativo generado por A es el cociente de A^* por la congruencia definida por las relaciones $ab = ba$ para toda letra a y b en A. Se escribe $A^{\oplus}$. El monoide $A^{\oplus}$ es isomorfo a $\mathbb{N}^k$, cuyos elementos son las k-tuplas de naturales $x = (x_1, x_2, ..., x_k)$. También es isomorfo a $a_1^* \times a_2^* \times \cdots \times a_k^*$, donde las a_i son letras del alfabeto A.*

Notación. *Escribimos e_i para referirnos al elemento de $\mathbb{N}^k$ cuyas componentes son todas 0 excepto la i-ésima, que es 1. Decimos que e_i es uno de los k generadores de $\mathbb{N}^k$.*

Generalizamos las expresiones regulares al contexto de $\mathbb{N}^k$. En este caso hablaremos de expresiones racionales, que se definen como uno esperaría.

Definición 2.4 (Expresión racional)**.** *Una expresión racional sobre $\mathbb{N}^k$ es una formula obtenida inductivamente a partir de elementos de $\mathbb{N}^k$ y las funciones $\{\emptyset, \cup, +, \oplus\}$ de la siguiente manera:*

I. *$\emptyset$ y cada elemento de $\mathbb{N}^k$ son expresiones racionales.*

II. *si E y F son expresiones racionales, luego $(E \cup F)$, $(E + F)$ y $(E^{\oplus})$ también son racionales.*

De manera similar a la Definición 1.6 se define el conjunto denotado por una expresión racional sobre $\mathbb{N}^k$.

Definición 2.5 (Conjunto denotado por una expresión racional)**.** *El conjunto denotado por una expresión racional E sobre $\mathbb{N}^k$, lo notamos $|E|$, se define inductivamente en la profundidad de la expresión:*

1. *Para las expresiones atómicas:*
   $L(\emptyset) = \emptyset$ y $L(c) = \{c\}$ para cualquier $c \in \mathbb{N}^k$.

2. *Para expresiones compuestas:*
   $L(E \cup F) = L(E) \cup L(F)$, $L(E + F) = L(E) + L(F)$ *y* $L(E^{\oplus}) = L(E)^{\oplus}$

Decimos que una expresión racional E denota al conjunto $|E|$.

Definición 2.6 (Conjunto racional)**.** *Un conjunto S de $\mathbb{N}^k$ es racional si está denotado por una expresión racional sobre $\mathbb{N}^k$.*

Usaremos $Rat(\mathbb{N}^k)$ para referirnos a la familia de conjuntos racionales.

Definición 2.7 (Altura estrella de una expresión racional)**.** *La altura estrella de una expresión racional E, a la que notamos $h[E]$, es la máxima cantidad de estrellas anidadas en la expresión E. Se define inductivamente para las expresiones racionales de $\mathbb{N}^k$,aunque se podría dar la definición para cualquier monoide:*

si $E = \emptyset$ *o* E *es un elemento de* $\mathbb{N}^k$	$h[E] = 0.$
si $E = F \cup G$ *o* $E = F + G$	$h[E] = \max(h[F], h[G]).$
si $E = F^{\oplus}$	$h[E] = 1 + h[F].$

Ejemplo 2.1. *La altura estrella de* $\big((0,1)^{\oplus} + (1,0)\big)^{\oplus}$ *es 2.*

Definición 2.8 (Altura estrella de un conjunto racional)**.** *La altura estrella de un conjunto racional S de $\mathbb{N}^k$, a la que notamos $h[S]$, es la mínima altura estrella de las expresiones racionales de $\mathbb{N}^k$ que denotan a S:*

$$h[S] = \min\{h[E] \ : \ E \text{ una expresión racional de } \mathbb{N}^k \text{ y } |E| = S\}$$

En el monoide $\mathbb{N}^k$, por ser conmutativo, cualquier conjunto racional se puede denotar con una expresión que tenga altura estrella a lo sumo 1 (ver libro [10, Exer. I.6.5]).

Proposición 2. *La altura estrella de $S \in Rat(\mathbb{N}^k)$ es a lo sumo 1.*

Ejemplo (Continuación del Ejemplo 2.1)**.** $\big((0,1)^{\oplus} + (1,0)\big)^{\oplus}$ *es equivalente a* $(1,0)^{\oplus} \ \cup \ (1,0) + \big((0,1) \cup (1,0)\big)^{\oplus}$ *que tiene altura estrella 1.*

Por lo tanto definimos expresiones donde restringimos la altura estrella:

Definición 2.9 (Expresión lineal y expresión semi-lineal)**.** *Una expresión E que denota un conjunto racional de $\mathbb{N}^k$ es:*

- *lineal si $E = c + B^{\oplus}$ donde $c \in \mathbb{N}^k$ y B es un conjunto finito de $\mathbb{N}^k$.*
- *semi-lineal si es la unión finita de expresiones lineales.*

Ejemplo (Continuación del Ejemplo 2.1)**.** $\{(1,0)\}^{\oplus} \ \cup \ (1,0) + \{(0,1),(1,0)\}^{\oplus}$ *es semi-lineal.*

Definición 2.10. *En $\mathbb{N}^k$ definimos las operaciones racionales no-ambiguas, una especialización de las operaciones racionales para las cuales usaremos los mismos símbolos:*

1. *Unión no-ambigua: $S \cup T$ es no-ambiguo si $S \cap T = \emptyset$*
2. *Suma no-ambigua: $S + T$ es no-ambigua si*
   $\forall s, s' \in S, \forall t, t' \in T \quad s + t = s' + t' \implies s = s' \ y \ t = t'$

3. *Estrella no-ambigua:* $S^{\oplus} = \bigcup_{n\in\mathbb{N}} \underbrace{S + \cdots + S}_{n \text{ veces}}$ *es no-ambigua si cada una de las sumas y uniones son no-ambiguas.*

Si $B = \{b_1, b_2, \ldots, b_l\}$, por conmutatividad del monoide vale lo siguiente:

$$B^{\oplus} = \{n_1 b_1 + n_2 b_2 + \cdots + n_l b_l \mid n_i \in \mathbb{N}\} = b_1^{\oplus} + b_2^{\oplus} + \cdots + b_l^{\oplus}.$$

Definición 2.11 (Base libre)**.** *Un conjunto* $B = \{b_1, b_2, \ldots, b_l\}$ *es una base libre si* $b_1^{\oplus} + b_2^{\oplus} + \cdots + b_l^{\oplus}$ *es una expresión racional no-ambigua, y esto quiere decir que la combinación lineal de sus elementos* $(n_1 b_1 + n_2 b_2 + \cdots + n_l b_l)$ *es siempre única, son linealmente independientes.*

Ejemplo 2.2. $B = \{(1,0), (3,1), (1,1)\}$ *no es libre pues* $2(1,0)+(1,1) = (3,1)$*. En cambio* $B' = \{(1,0), (1,1)\}$ *es una base libre.*

Definición 2.12 (Expresión simple y expresión semi-simple)**.** *Una expresión* E *que denota a un conjunto de* $\mathbb{N}^k$ *es:*

- *simple si* E *es lineal* $(E = c + B^{\oplus})$ *y* B *es una base libre.*
- *semi-simple si es la unión no-ambigua de expresiones simples.*

Ejemplo (Continuación del Ejemplo 2.1)**.** $\{(1,0)\}^{\oplus} \cup (1,0) + \{(0,1),(1,0)\}^{\oplus}$ *no es semi-simple. Ambas expresiones,* $\{(1,0)\}^{\oplus}$ *y* $(1,0) + \{(0,1),(1,0)\}^{\oplus}$*, son simples porque ambas bases son libres. Sin embargo, su unión no es disjunta, ya que* $[\{(1,0)\}^{\oplus}] \cap [(1,0) + \{(0,1),(1,0)\}^{\oplus}] = (1,0)+(1,0)^{\oplus}$*. Una expresión semi-simple equivalente es:* $\{(1,0)\}^{\oplus} \cup (1,1) + \{(0,1),(1,0)\}^{\oplus}$

Proposición 3 ([3, Teorema])**.** *Cualquier conjunto racional de* $\mathbb{N}^k$ *puede ser denotado con una expresión semi-simple.*

Comentario. *Hay una versión algorítmica de la Proposición 3 en el libro de Sakarovitch [10, Sec. II.7.5].*

Por ultimo trabajaremos con series formales de k variables conmutativas. En particular nos interesa la serie característica de conjuntos de $\mathbb{N}^k$.

Definición 2.13 (Serie característica de un conjunto racional de $\mathbb{N}^k$)**.** *Sea* S *un conjunto racional de* $\mathbb{N}^k$*. Definimos* $\underline{S} \in \mathbb{Z}\langle\langle \mathbb{N}^k \rangle\rangle$*, su serie característica, a partir de una expresión semi-simple de* S*. Siendo* $c = (c_1, c_2, \ldots, c_k)$ *un elemento de* $\mathbb{N}^k$ *y* $B = \{b_1, b_2, \ldots, b_l\}$ *una base libre:*

- $\underline{c} = x_1^{c_1} x_2^{c_2} \ldots x_k^{c_k}$
- $\underline{B^{\oplus}} = \dfrac{1}{(1-\underline{b_1})(1-\underline{b_2})\cdots(1-\underline{b_l})}$
- $\underline{c + B^{\oplus}} = \underline{c}\ \underline{B^{\oplus}}$
- $\underline{E \cup F} = \underline{E} + \underline{F}$ *siempre que la unión sea disjunta.*

Ejemplo 2.3. *Consideremos S_a denotado por $(1,1)+\{(1,1)\}^{\oplus}$. Entonces, $\underline{S_a} = \dfrac{xy}{(1-xy)}$.*

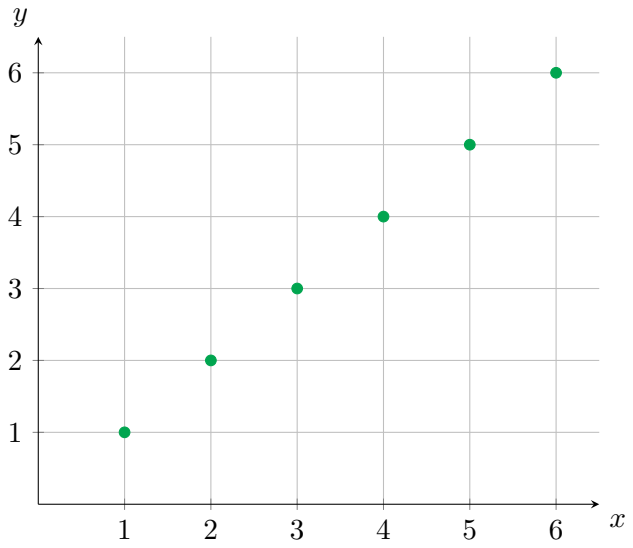

Figura 2.1: $(1,1)+\{(1,1)\}^{\oplus}$

Como ya mencionamos, no todo lenguaje regular $L \subseteq A^*$ tiene clausura conmutativa $\mathcal{C}(L)$ regular, como subconjunto de A^*. Sin embargo, su imagen conmutativa es siempre racional.

Lema 4. *Sea L un lenguaje regular en A^*, su imagen conmutativa $\varphi(L)$ es un conjunto racional en $\mathbb{N}^k$.*

Por lo tanto, que un conjunto S sea racional en $\mathbb{N}^k$ no nos dice nada acerca de si $\varphi^{-1}(S)$ es o no regular.

Otra familia interesante que se define y estudia en monoides finitamente generados, son los conjuntos reconocibles. Nuestro interés proviene de que la imagen inversa por cualquier morfismo suryectivo de un conjunto reconocible en $\mathbb{N}^k$ es un conjunto regular en A^*.

Definición 2.14 (Conjunto reconocible)**.** *Sea A un alfabeto finito y sea $\varphi : A^* \to \mathbb{N}^k$ la imagen conmutativa. Un subconjunto S de $\mathbb{N}^k$ es reconocible si $\varphi^{-1}(S)$ es regular. Usaremos $Rec(\mathbb{N}^k)$ para denotar a la familia de conjuntos reconocibles de $\mathbb{N}^k$.*

La siguiente es una definición equivalente de los conjuntos reconocibles que se debe a Mezei.

Definición 2.15 (Mezei [10, Cor. II.2.20])**.** *Un conjunto S de $\mathbb{N}^k$ es reconocible si existe una familia de conjuntos racionales $T_{i1}, \ldots, T_{ik} \subseteq \mathbb{N}$ con i indexado sobre un conjunto finito I tales que*

$$S = \bigcup_{i \in I} T_{i_1} \times \cdots \times T_{i_k}.$$

Comentario. *Esta expresión para definir el conjunto S no es única. De hecho, en el caso mas simple de todos, que es cuando S es $\mathbb{N}$, ya nos encontramos con múltiples expresiones:*

- $1^{\oplus}$

- $1 \cup 1 + 1^{\oplus}$
- $1 \cup \cdots \cup n \cup (n+1) + 1^{\oplus}$
- $2^{\oplus} \cup 1 + 2^{\oplus}$
- ...

El algoritmo de Gohon se basa en los resultados que permiten demostrar el siguiente teorema:

Proposición 5 (Gohon[5, Theorem 4.6]). *Para todo conjunto $S \in Rat(\mathbb{N}^k)$, podemos asociar una fracción $\underline{S} = P/Q$, donde P y Q son polinomios de las variables conmutativas $x_1, x_2, ..., x_k$ con coeficientes en $\mathbb{Z}$, tal que S es reconocible si y sólo si Q es un producto de polinomios de una sola variable.*

Para demostrar el teorema de la Proposición 5, Gohon prueba los resultados de las Proposiciones 5,6 y 7.

Proposición 6 (Gohon[5]). *Sea $S \in Rat(\mathbb{N}^k)$, Luego existe un único par de polinomios $P(S)$ y $Q(S) \in \mathbb{Z}[x_1, x_2, ..., x_k]$ tales que:*

I. $\underline{S} = P(S)/Q(S)$.

II. $P(S)/Q(S)$ *es irreducible.*

III. $Q(S)[1] = 1$, *es decir el coeficiente del termino independiente es 1.*

Más aún, para todo par de polinomios P y $Q \in \mathbb{Z}[x_1, x_2, ..., x_k]$ tales que $\underline{S} = P/Q$, existe un polinomio $R \in \mathbb{Z}[x_1, x_2, ..., x_k]$ tal que $P = R.P(S)$ y $Q = R.Q(S)$.

Comentario. *El hecho de que los polinomios de la serie característica sean únicos para un conjunto racional, como sucede en $\mathbb{N}^k$, no vale en cualquier otro monoide. Por ejemplo, en A^* no son únicos. Esta propiedad es fundamental para probar la Proposición 5 ya que en $\mathbb{N}^k$ la serie característica nos da una representación canónica de los conjuntos racionales.*

Además cuando S es reconocible la Proposición 5 nos dice que los polinomios de la serie característica asociados a S van a tener la siguiente pinta.

Proposición 7 (Gohon [5]). *Sea $S \in Rat(\mathbb{N}^k)$. Si $S \in Rec(\mathbb{N}^k)$, $P(S)$ y $Q(S)$ en $\mathbb{Z}[x_1, x_2, ..., x_k]$ (de la Proposición 6) entonces $Q(S) = 1$, o $Q(S)$ es un producto de polinomios de la forma $(1 - x_j^{\alpha_j})$.*

El algoritmo de Gohon parte de una expresión semi-simple y obtiene su serie característica (ver Definición 2.13). Reduce los polinomios hasta que todos los factores de más de una variable en el denominador se simplifiquen. Esto siempre es posible cuando el conjunto es reconocible. Por lo tanto se obtienen dos polinomios que cumplen la siguiente proposición.

Proposición 8 ([5, 4.5]). *Sea $S \in Rat(\mathbb{N}^k)$ y sean dos polinomios P y $Q \in \mathbb{Z}[x_1, x_2, ..., x_k]$ tales que:*

(i) $\underline{S} = P/Q$,

(ii) $Q = 1$ *o* Q *es un producto de polinomios de la forma* $(1 - x_j^{\alpha_j})$ $(\alpha_j \in \mathbb{N}_{>0})$.

Luego, $S \in Rec(\mathbb{N}^k)$.

Ejemplo (Continuación de Ejemplo 2.3)**.** $\underline{S_a} = \dfrac{xy}{(1-xy)}$

El conjunto S_a *no es reconocible: notemos que* $\underline{S_a}$ *es irreducible y tiene mas de una variable en el denominador. De hecho este conjunto* S_a *se corresponde con la clausura conmutativa de* $(ab)^+$ *si consideramos el alfabeto* $A = \{a, b\}$

Ejemplo 2.4. $S_b = (0,1) + \{(2,0),(0,2)\}^{\oplus}$, $\underline{S_b} = \dfrac{y}{(1-x^2)(1-y^2)}$.

El conjunto S_b *es reconocible: notemos que* $\underline{S_b}$ *no tiene factores de más de una variable en el denominador. Este conjunto es la imagen conmutativa del lenguaje* $L = |b(a^2 \cup b^2)^*|$ *del Ejemplo 1.1, para el que ya sabíamos que su clausura conmutativa es regular porque dimos el autómata finito que lo acepta.*

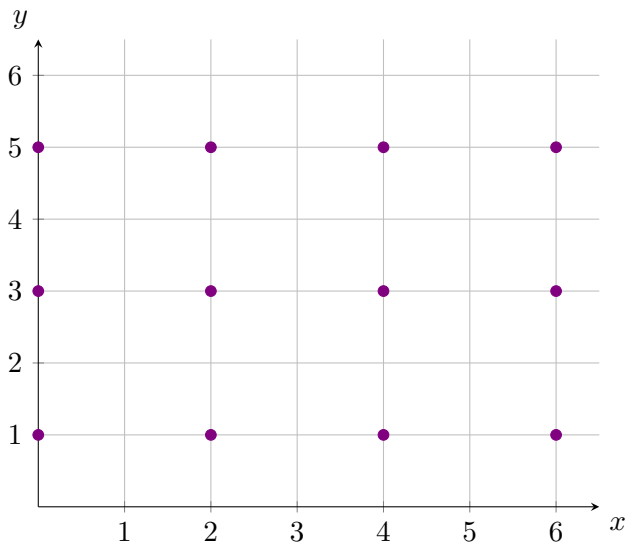

Figura 2.2: $(0,1) + \{(2,0),(0,2)\}^{\oplus}$

3. LAS EXPRESIONES CARACOLADAS

En este capitulo definimos unas expresiones para nombrar a los conjuntos reconocibles de $\mathbb{N}^k$ y que podremos construir a partir de la serie característica.

Definición 3.1 (Elemento primario)**.** *Un elemento $b \in \mathbb{N}^k$ es **primario** si $b = n\, e_j$ para un $n \in \mathbb{N}_{>0}$ y un j entre 1 y k.*

Cuando queremos explicitar el índice j diremos que b es j-primario.

Definición 3.2 (Base primaria)**.** *Una base libre B es **primaria** si todos sus elementos son primarios.*

Definición 3.3 (Expresión caracolada atómica)**.** *Una expresión C de un conjunto (reconocible) de $\mathbb{N}^k$ es **caracolada atómica** si C es simple, por lo tanto $C = a + B^{\oplus}$, y además B es una base primaria.*

Una expresión caracolada atómica de $\mathbb{N}$, la llamaremos expresión caracola (ChAracterization of RAtionals in the Case of One-Letter Alphabets).

Una expresión caracolada atómica es una expresión racional de $\mathbb{N}^k$. El conjunto denotado por una expresión caracolada atómica C, lo notamos $|C|$ y coincide con la Definición 1.6 que dimos para expresiones racionales de $\mathbb{N}^k$.

A partir de una expresión caracola $C = \gamma + \alpha^{\oplus}$, donde $\alpha, \gamma \in \mathbb{N}$ podemos construir un AFD $\mathcal{A} = \langle Q, \{a\}, E, I, T\rangle$ que reconoce el lenguaje $\varphi^{-1}(|C|) \subseteq a^*$, donde φ es la imagen conmutativa (Definición 1.9).

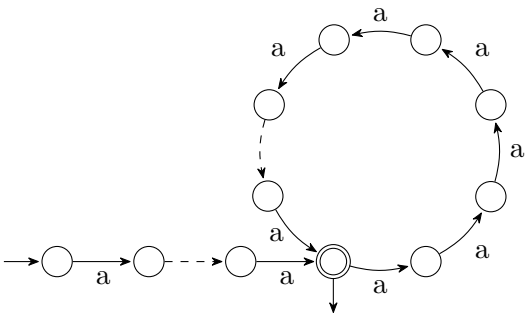

Figura 3.1: AFD de $\varphi^{-1}(|C|)$ cuando C es una expresión caracola

.

Para el caso $k = 1$ ya conseguimos dar el autómata, ahora buscamos hacer lo mismo para cualquier k. Pero antes debemos regresar a A^* para introducir una operación entre lenguajes regulares.

Definición 3.4 (Mezcla)**.** *La mezcla de dos palabras w y v de A^*, $w \between v$, es el conjunto de A^* definido por:*

$$w \between v = \{w_1 v_1 \cdots w_n v_n \mid w = w_1 \cdots w_n, v = v_1 \cdots v_n, w_i, v_j \in A^*\}.$$

La mezcla de dos palabras se extiende aditivamente a $\mathcal{P}(A^*)$*,*

$$L \between K = \bigcup_{w \in L, v \in K} w \between v.$$

Comentario. *La operación de la mezcla que acabamos definir se la conoce comúnmente como shuffle en ingles. Optamos por usar* $\between$ *para el operador aunque este no es estándar.*

Ejemplo 3.1. $ab \between ba = \{abba, baba, baab, abab\}$.

Proposición 9. *La mezcla de dos lenguajes regulares en* A^* *es regular y por lo tanto se puede dar un autómata finito que la reconoce.*

Demostración. Sea $\mathcal{A}' = \langle Q', A, E', I', T' \rangle$ y $\mathcal{A}'' = \langle Q'', A, E'', I'', T'' \rangle$.
Definimos $\mathcal{A} = \mathcal{A}' \between \mathcal{A}''$ como $\mathcal{A} = \langle Q' \times Q'', A, E, I' \times I'', T' \times T'' \rangle$, con E:

$$\begin{aligned} E = &\{((p', p''), a, (q', p'')) | p'' \in Q'' \text{ y } ((p', a, q') \in E'\} \cup \\ &\{((p', p''), a, (p', q'')) | p' \in Q' \text{ y } ((p'', a, q'') \in E''\} \end{aligned}$$

Es fácil ver que $|\mathcal{A}| = |\mathcal{A}'| \between |\mathcal{A}''|$. □

Definición 3.5. *Al autómata* $\mathcal{A}' \between \mathcal{A}''$ *que utilizamos en la prueba anterior lo llamamos producto mezcla entre los autómatas* $\mathcal{A}'$ *y* $\mathcal{A}''$ *y es una construcción general que nos permite obtener un autómata para reconocer la mezcla de dos lenguajes a partir de dos autómatas que reconocen cada uno de esos lenguajes.*

Inductivamente, dados AFDs $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *es posible definir un autómata* $\mathcal{A} = \mathcal{A}_1 \between \cdots \between \mathcal{A}_n$ *que reconoce la mezcla entre los lenguajes reconocidos por todos ellos.*

Proposición 10. *Sean* A' *y* A'' *alfabetos disjuntos, y* $A = A' \cup A''$ *la unión de ambos. Supongamos que tenemos dos AFDs* $\mathcal{A}'$ *y* $\mathcal{A}''$ *definidos sobre los alfabetos* A' *y* A'' *respectivamente. Si consideramos a ambos autómatas como autómatas sobre A, entonces el producto mezcla entre ellos* $\mathcal{A}' \between \mathcal{A}''$ *es también un AFD.*

Demostración. Por el absurdo, supongamos que $\mathcal{A}' \between \mathcal{A}''$ no es determinístico. Claramente el autómata que construimos tiene un solo estado inicial para que $|I' \times I''| > 1$, tendríamos $|I'| > 1$ o $|I''| > 1$ pero estos eran determinísticos por hipótesis. Entonces tiene que haber mas de una transición con algún origen (p', p'') y etiqueta a en E. Como los alfabetos son disjuntos hay dos posibilidades:

- $a \in A'$ *y* $a \notin A''$ como no hay transiciones en E'' con a ambas deben estar en E'. Por lo que hay mas de una transición en E' con origen p' y etiqueta a. Absurdo, $\mathcal{A}'$ es determinístico.
- $a \in A''$ *y* $a \notin A'$ que es análogo al anterior y también llegamos a una contradicción, que $\mathcal{A}''$ es determinístico en este caso.

□

Ahora sí, construimos el autómata en A^* para cualquier expresión caracolada atómica así como lo hicimos con $k = 1$.

Proposición 11. *Sea $C = (\gamma_1, \ldots, \gamma_k) + B^{\oplus}$ una expresión caracolada atómica que denota un conjunto $S \subseteq \mathbb{N}^k$. Podemos construir un AFD $\mathcal{A}$ que acepta el lenguaje $\varphi^{-1}(S) \subseteq A^*$, donde φ es la imagen conmutativa (Definición 1.9).*

Demostración. Como los elementos de B son primarios podemos escribir $S = S_1 \times \cdots \times S_k$, donde cada S_j es denotado por una expresión caracola $C_j = \gamma_j + \alpha_j^{\oplus}$. Para cada j podemos obtener un AFD completo $\mathcal{A}_j$ etiquetado sobre el alfabeto $\{a_j\}$ que reconoce $\varphi^{-1}(S_j)$. (Ver Figura 3.1).

Pensando a los autómatas $\mathcal{A}_j$ como etiquetados en el alfabeto más grande $A \supseteq \{a_j\}$ podemos considerar el producto mezcla

$$\mathcal{A} = \mathcal{A}_1 \between \cdots \between \mathcal{A}_k$$

Este autómata reconoce el lenguaje $\varphi^{-1}(S)$. Para convencerse de ello, basta observar que $w \in |\mathcal{A}|$ si y sólo si para todo j vale que $|w|_{a_j} \in S_j$. Y esto último es equivalente a que $\varphi(w) \in S$. □

Ejemplo 3.2. *Expresión caracolada atómica y su autómata.*

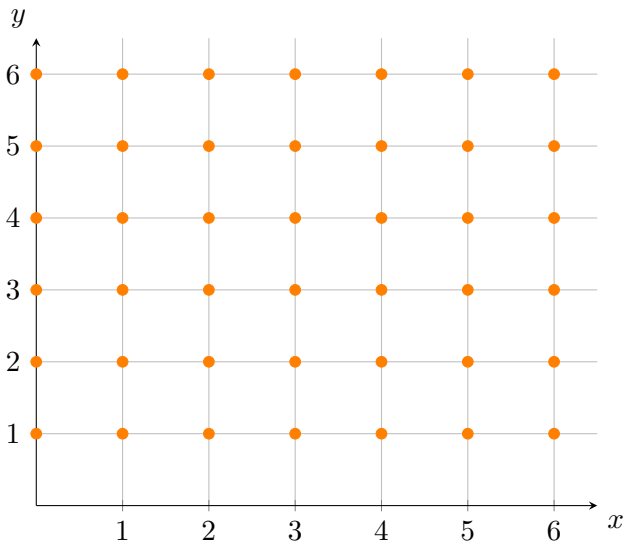

Figura 3.2: $(0,1) + \{(1,0),(0,1)\}^{\oplus}$

Construimos el automata usando la Proposición 11. Las expresiones caracolas son $C_1 = 1^{\oplus}$ *y* $C_2 = 1 + 1^{\oplus}$

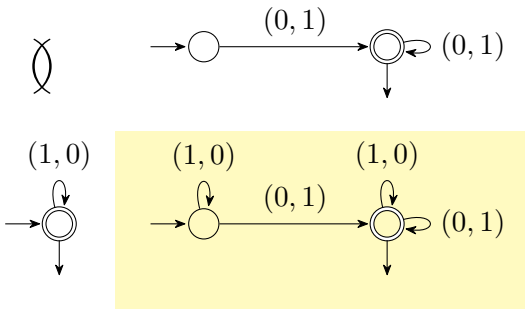

Figura 3.3: Construcción del autómata de $\varphi^{-1}(|(0,1) + \{(1,0),(0,1)\}^{\oplus}|)$ en $\{(0,1),(1,0)\}^*$. Sombreado el autómata resultante

Comentario. *Todos los autómatas finitos que daremos en los ejemplos serán sobre G^*, donde $G = \{e_1, \ldots, e_k\}$ es el conjunto de los generadores de $\mathbb{N}^k$ considerado como un alfabeto. De ahora en más, diremos el autómata de una expresión caracolada C para referirnos al autómata que acepta a $\varphi^{-1}(|C|)$, donde $\varphi : G^* \to \mathbb{N}^k$ es la imagen conmutativa. Ademas, por claridad, todos los ejemplos son en $\mathbb{N}^2$.*

También, es posible calcular el tamaño exacto del autómata construido en la Proposición 11.

Proposición 12. *Sea una expresión caracolada atómica $C = (\gamma_1, \ldots, \gamma_k) + B^{\oplus}$. Para cada j, consideramos α_j definido como*

$$\alpha_j = \begin{cases} b_j & \text{si } b_j \text{ es el período del único elemento } j-\text{primario de } B \text{ si es que existe} \\ 2 & \text{si no existe elemento } j-\text{primario en } B \end{cases}$$

Entonces, la cantidad de estados del AFD completo definido en la Proposición 11 y que reconoce $\varphi^{-1}(|C|)$ es exactamente

$$\prod_{j=1}^{k} \gamma_j + \alpha_j.$$

Comentario. *El caso en que no existe elemento $j-$primario en B la caracola de esa componente que definimos en la Proposición 11 es finita, $C_j = \gamma_j$. Por lo tanto, el AFD de C_j tendría $\gamma_j + 1$ estados. Pero para que sea **completo** es necesario agregar un estado trampa, teniendo $\gamma_j + 2$ estados en total. Es por eso que en este caso $\alpha_j = 2$.*

Ejemplo 3.3. *Otro ejemplo de expresión caracolada atómica y su autómata.*

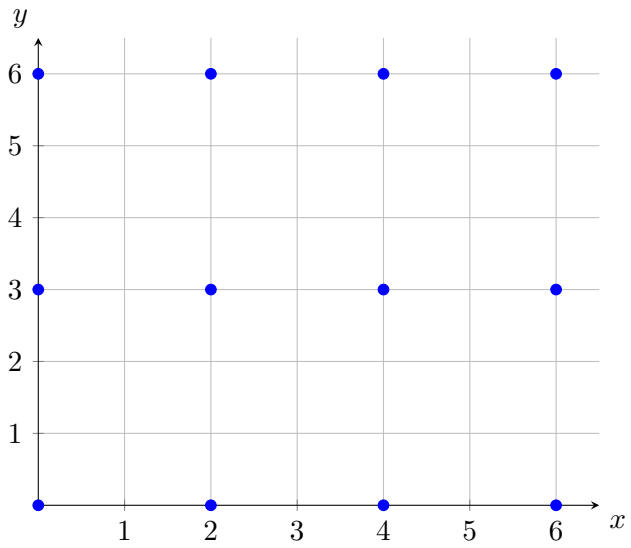

Figura 3.4: $\{(2,0),(0,3)\}^{\oplus}$

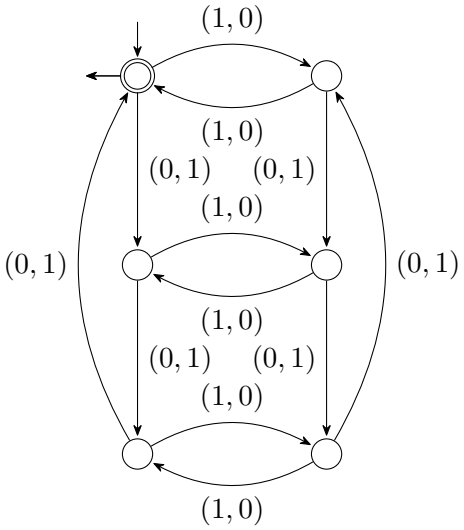

Figura 3.5: Autómata de $\varphi^{-1}(|\{(2,0),(0,3)\}^{\oplus}|)$

Otro ejemplo de expresión caracolada atómica es el Ejemplo 1.1 del capitulo anterior.

Definición 3.6 (Expresión caracolada de un conjunto de $\mathbb{N}^k$)**.** *Una expresión caracolada de un conjunto (reconocible) de $\mathbb{N}^k$ es una formula que se obtiene inductivamente a partir de expresiones caracoladas atómicas y las operaciones booleanas$(\cup,\cap,^c)$:*

I. *Una expresión caracolada atómica es caracolada.*

II. *Si E, F son expresiones caracoladas entonces $E \cup F$, $E \cap F$ y E^c son caracoladas.*

Ejemplo 3.4. *Consideramos el conjunto reconocible $S_C \subseteq \mathbb{N}^2$,*

$$S_C = |(0,1) + \{(1,0),(0,1)\}^{\oplus} \cup \{(2,0),(0,3)\}^{\oplus}|$$

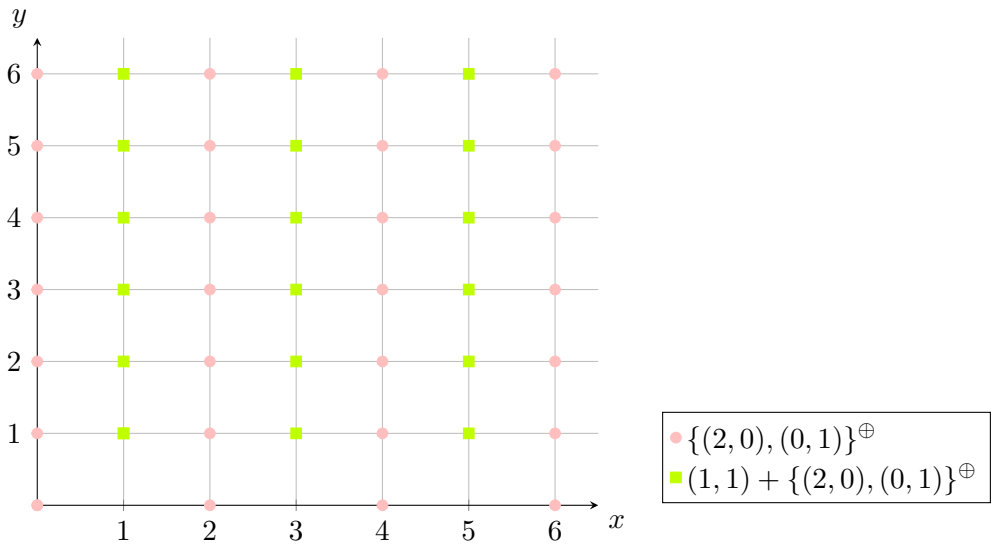

Figura 3.6: Conjunto reconocible $S_c = |(0,1) + \{(1,0),(0,1)\}^{\oplus} \cup \{(2,0),(0,3)\}^{\oplus}|$

Proposición 13. *Sea S un conjunto de $\mathbb{N}^k$. S tiene una expresión caracolada que lo denota entonces $S \in Rec(\mathbb{N}^k)$.*

Demostración. Porque $Rec(\mathbb{N}^k)$ es un álgebra Booleana y las expresiones caracoladas atómicas denotan conjuntos reconocibles. □

Definición 3.7 (Expresión caracolada consistente)**.** *Una expresión es caracolada **consistente** si es caracolada y todas las bases de sus átomos son iguales.*

Ejemplo (Continuación del Ejemplo 3.4)**.** *La expresión caracolada que dimos antes para denotar S_c: $(0,1)+\{(1,0),(0,1)\}^{\oplus} \cup \{(2,0),(0,3)\}^{\oplus}$, no es consistente pues $\{(1,0),(0,1)\} \neq \{(2,0),(0,3)\}$. Pero podemos dar una expresión equivalente que lo es: $C_c = \{(2,0),(0,1)\}^{\oplus} \cup (1,1) + \{(2,0),(0,1)\}^{\oplus}$*

Teorema 2. *Sea C una expresión caracolada que denota un conjunto $S \in Rec(\mathbb{N}^k)$. Podemos construir un AFD para el lenguaje $\varphi^{-1}(S) \subseteq A^*$, donde φ es la imagen conmutativa (Definición 1.9).*

Demostración. Consecuencia inmediata del Proposición 11 y el hecho de que los conjuntos regulares forman un álgebra booleana efectiva. □

Proposición 14. *La intersección de dos lenguajes regulares en A^* es regular y se puede dar un autómata finito que la reconoce.*

Demostración. Sea $\mathcal{A}' = \langle Q', A, E', I', T' \rangle$ y $\mathcal{A}'' = \langle Q'', A, E'', I'', T'' \rangle$.

Definimos $\mathcal{A} = \mathcal{A}' \cap \mathcal{A}''$ como $\mathcal{A} = \langle Q' \times Q'', A, E, I' \times I'', T' \times T'' \rangle$, con E:

$$E = \{((p', p''), a, (q', q'')) | ((p', a, q') \in E' \text{ y } ((p'', a, q'') \in E''\}.$$

Es fácil ver que $|\mathcal{A}| = |\mathcal{A}'| \cap |\mathcal{A}''|$. □

Al autómata $\mathcal{A}' \cap \mathcal{A}''$ que utilizamos en la prueba anterior lo llamamos *producto intersección* entre los autómatas $\mathcal{A}'$ y $\mathcal{A}''$ y es una construcción general que nos permite obtener un autómata para reconocer la intersección de dos lenguajes a partir de dos autómatas que reconocen cada uno de esos lenguajes. La siguiente Proposición es una verificación de rutina.

Proposición 15. *Si $\mathcal{A}'$ y $\mathcal{A}''$ son AFDs entonces el producto intersección $\mathcal{A} = \mathcal{A}' \cap \mathcal{A}''$ también es un AFD.*

Nos interesa ver que los autómatas definidos en la Proposición 11 a partir de expresiones caracoladas atómicas no crecen significativamente en tamaño al intersecarlos. Para ello probaremos la siguiente Proposición.

Proposición 16. *Si $\mathcal{A}'$ y $\mathcal{A}''$ provienen de expresiones caracoladas consistentes entre si, la intersección $\mathcal{A} = \mathcal{A}' \cap \mathcal{A}''$ no aumenta la cantidad de estados (accesibles) máxima por cada dimensión.*

Demostración. Dado que los autómatas en cada dimensión terminan con el mismo periodo siempre al autómata mas grande para una cierta letra le va a corresponder un único estado en el otro autómata. Es decir en cada dimensión se terminan emparejando los autómatas, ver Figura 3.7. □

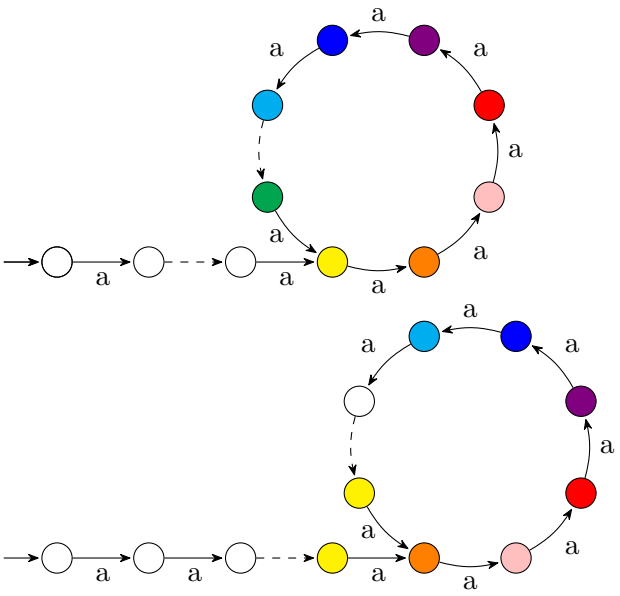

Figura 3.7: Emparejamiento de dos AFDs de caracoladas consistentes con $k = 1$

Comentario. *Si en una dimensión alguno de los autómatas era completo luego el resultado de la intersección sera completo en esa dimensión. Si no lo fuera bastaría con agregar un estado trampa.*

Ejemplo (Continuación del Ejemplo 3.4)**.** *Construcción del autómata de $|C_c|$*

$$C_c = \{(2,0),(0,1)\}^{\oplus} \cup (1,1) + \{(2,0),(0,1)\}^{\oplus}$$

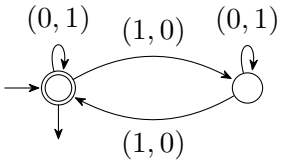

Figura 3.8: Autómata de $\{(2,0),(0,1)\}^{\oplus}$

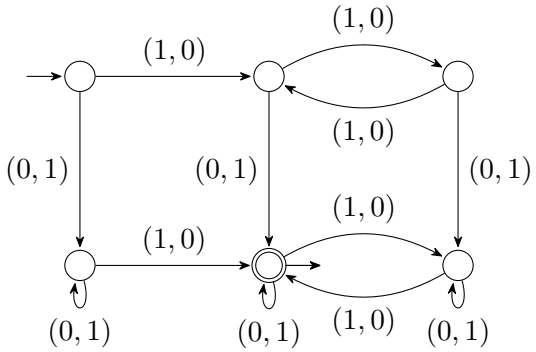

Figura 3.9: Autómata de $(1,1) + \{(2,0),(0,1)\}^{\oplus}$

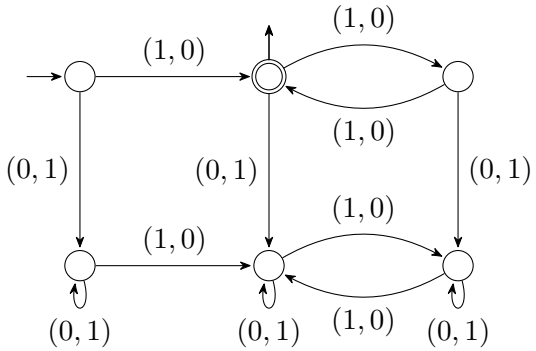

Figura 3.10: Autómata de $C'_c = \{(2,0),(0,1)\}^{\oplus^c} \cap ((1,1) + \{(2,0),(0,1)\}^{\oplus})^c$

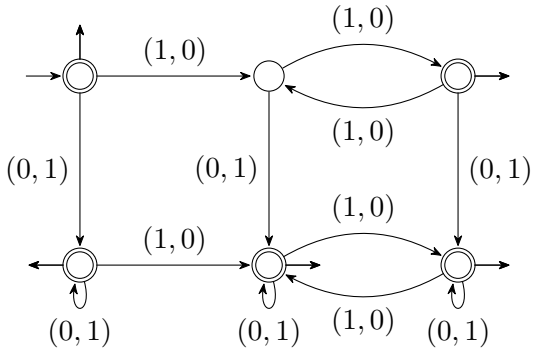

Figura 3.11: Autómata de C_c, el complemento de Figura 3.10

4. DEL POLINOMIO A LA EXPRESIÓN CARACOLADA

En este capitulo damos un método efectivo para obtener una expresión caracolada que denote el conjunto, partiendo de los polinomios de su serie característica.

Recordemos que el algoritmo de Gohon convierte la expresión semi-simple en una serie característica que expresa como una fracción de polinomios P_0 y Q_0. Luego reduce esta fracción simplificando todos los factores de más de una variable en Q_0, lo cual se puede hacer si sólo si el conjunto denotado es reconocible. Estos polinomios P y Q que ya se simplificaron cumplen con las condiciones de la Proposición 8. Es decir Q es un producto de polinomios de la forma $(1 - x_j^{\alpha_j})$ $(\alpha_j \in \mathbb{N}_{>0})$ o 1.

También, se puede garantizar que para toda j de $\{1, \ldots, k\}$ hay a lo sumo un factor de la forma $(1 - x_j^{\alpha_j})$ gracias al siguiente lema.

Lema 17 (Gohon[5, 4.3])**.** *Sea $S \in Rat(\mathbb{N}^k)$. Existe una expresión semi-simple que denota S y que para cada componente j se verifican las dos condiciones siguientes:*

- *Existe a lo sumo un elemento j-primario en cada base de la expresión*
- *Si dos bases en la expresión contienen un elemento j-primario, entonces es el mismo en ambas.*

Además, la demostración de Gohon da un procedimiento efectivo para calcular la expresión semi-simple a partir de una cualquiera. Tomando para cada j-ésima componente el mínimo común múltiplo de todas las j-ésimas componentes de los generadores j-primarios de la expresión. Luego basta con convertir a serie e ir simplificando los factores de mas de una variable. Esto Gohon lo utiliza y prueba para la demostración de Proposición 8, nosotros lo enunciamos como un lema auxiliar:

Lema 18. *Sea $S \in Rec(\mathbb{N}^k)$, podemos computar P_0 y Q_0 dos polinomios tal que:*

(i) $\underline{S} = P_0/Q_0$,

(ii) $Q_0 = 1$ *o* $(\exists J \subseteq \{1, 2, \ldots, k\})(Q_0 = \prod_{j \in J}(1 - x_j^{\alpha_j})(\alpha_j \in \mathbb{N} \setminus \{0\}))$.

Demostración. Sea E una expresión semi-simple de S. Por Lema 17 podemos calcular una equivalente, E', tal que por cada $j \in \{1, 2, \ldots, k\}$ existe a lo sumo un generador j-primario. A partir de E' calculamos $\underline{S}$ usando la Definición 2.13. Obtenemos dos polinomios P_1 y Q_1 tal que $\underline{S} = P_1/Q_1$ y Q_1 es un producto de polinomios de la forma $(1 - x_1^{\beta_1} \cdots x_k^{\beta_k})$. Para cada j de $\{1, \ldots, k\}$ existe a lo sumo un polinomio de la forma $(1 - x_j^{\alpha_j})$ y por Proposición 7 podemos deducir que P_1 es divisible por los polinomios de más de una variable que constituyen Q_1. Luego de esta simplificación obtenemos P_0 y Q_0. □

Ahora, planteamos una proposición que a partir de los mismos polinomios de la Proposición 8 P_0 y Q_0 $(\underline{S} = P_0/Q_0)$ nos da una expresión caracolada consistente que denote a S.

Teorema 3. *Sea $S \in Rec(\mathbb{N}^k)$ podemos dar una expresión caracolada consistente para S.*

La prueba es similar a la propuesta por Gohon para la Proposición 8

Demostración. A partir de una semi-simple de S utilizando el Lema 18 obtenemos dos polinomios P y Q.

Si $Q = 1$ entonces S es finito y podemos calcular una expresión caracolada que lo denote aplicando directamente la inversa de Definición 2.13.

Teniendo en cuenta que según la definición $x_1^{\delta_1} x_2^{\delta_2} \cdots x_k^{\delta_k} = \underline{(\delta_1, \delta_2, \ldots, \delta_k)}$, el polinomio P se tiene que poder escribir como

$$P = \sum_{1 \leq h \leq m} \underline{d_h}$$

Luego, basta con convertir cada término de la sumatoria finita $\underline{d_h}$ en su correspondiente d_h y unirlos. La expresión caracolada en este caso queda:

$$\bigcup_{1 \leq h \leq m} d_h.$$

De ahora en más asumimos que $Q \neq 1$. Sin pérdida de generalidad, asumimos para simplificar la notación $J = \{1, 2, \ldots, k'\}$ con $1 \leq k' \leq k$. En este caso otra vez usamos la Definición 2.13 y podemos escribir a P como:

$$P = \sum_{1 \leq h \leq m} \mu_h \underline{d_h}.$$

con $\mu_h \neq 0$ y $d_h \in \mathbb{N}^k$ para todo $(1 \leq h \leq m)$. Observar que a diferencia del caso finito acá puede haber términos negativos.

Para cada $j \in J$, definimos el elemento primario $b_j = \alpha_j \, e_j$ donde los α_j coinciden con los exponentes de Q. Además, para cada h $(1 \leq h \leq m)$, denotamos $S_h = d_h + b_1^{\oplus} + b_2^{\oplus} + \cdots + b_{k'}^{\oplus}$. A partir de estas definiciones tenemos

$$\underline{S_h} = \frac{d_h}{\underline{Q}}.$$

Luego,

$$\underline{S} = \sum_{1 \leq h \leq m} \mu_h \underline{S_h}.$$

Consideremos la clase de equivalencia $\sim$ definida en $\mathbb{N}^k$ como

$$s \sim s' \iff \forall h \ (1 \leq h \leq m) : \ s \in S_h \iff s' \in S_h.$$

Notemos que S_h son caracoladas, y además tenemos

$$s \sim s' \implies \forall h \ (1 \leq h \leq m) \ \underline{S_h}(s) = \underline{S_h}(s').$$

Luego,

$$s \sim s' \implies \underline{S}(s) = \underline{S}(s').$$

El conjunto de elementos s de $\mathbb{N}^k$ tal que $\underline{S}(s) = 1$ es una unión de clases de equivalencia. Entonces, S es una unión de caracoladas. Para ver que es caracolada faltaría probar que esa unión es finita.

A continuación, vemos que es finita y damos concretamente la expresión caracolada para S. Recordemos que por definición de la serie teníamos:

$$\underline{S}(s) = \begin{cases} 1 & \text{si } s \in S \\ 0 & \text{en otro caso.} \end{cases}$$

Esto ya lo usamos cuando consideramos sólo las clases de equivalencia cuyos elementos cumplían $\underline{S}(s) = 1$. Pero también podemos aplicarlo sobre los S_h. Por lo que para cada h $(1 \leq h \leq m)$ y elemento de la clase de equivalencia tenemos

$$y_h^{[s]} = \underline{S_h}(s) \in \{0, 1\}.$$

Ahora bien como sólo estamos interesados en las clases de equivalencia que:

$$\begin{aligned} \underline{S}([s]) &= 1 \\ \sum_{1 \leq h \leq m} \mu_h \underline{S_h}([s]) &= 1 \\ \sum_{1 \leq h \leq m} \mu_h y_h^{[s]} &= 1 \\ \mu_1 y_1^{[s]} + \cdots + \mu_m y_m^{[s]} &= 1. \end{aligned}$$

Luego solo nos interesan los y del conjunto

$$Y = \{(y_1, \ldots, y_m) \in \{0, 1\}^m | \mu_1 y_1 + \cdots + \mu_m y_m = 1\}.$$

Cada una de estas tuplas $y = (y_1, \ldots, y_m)$ representa a una clase de equivalencia por como fue definida. Ahora podemos dar una expresión caracolada a su clase de equivalencia asociada

$$C_y = T_1 \cap T_2 \cap \cdots \cap T_m \text{ con } T_h = \begin{cases} S_h & \text{si } y_h = 1 \\ (S_h)^c & \text{si } y_h = 0. \end{cases}$$

Finalmente estamos en condiciones de dar la expresión caracolada de S:

$$\bigcup_{y \in Y} C_y.$$

Ademas, esta expresión caracolada es consistente. Las caracoladas atómicas (S_h) tienen todas la misma base $B = \{b_1, b_2, \ldots, b_{k'}\}$. □

Ejemplo 4.1. *Consideremos S el conjunto reconocible denotado por la expresión semi-simple:*

$$E = (1,1) + \{(1,1)\}^{\oplus} \cup (1,0) + \{(1,0),(1,1)\}^{\oplus} \cup (0,1) + \{(0,1),(1,1)\}^{\oplus}$$

Su serie característica:

$$\begin{aligned} \underline{S} &= \frac{xy}{(1-xy)} + \frac{x}{(y-xy)(1-x)} + \frac{y}{(1-xy)(1-y)} \\ &= \frac{xy(1-x)(1-y) + x(1-y) + y(1-x)}{(1-xy)(1-x)(1-y)} \\ &= \frac{xy - x^2y - xy^2 + x^2y^2 + x - xy + y - xy}{(1-xy)(1-x)(1-y)} \\ &= \frac{(1-xy)(x+y-xy)}{(1-xy)(1-x)(1-y)} \\ &= \frac{x+y-xy}{(1-x)(1-y)} \end{aligned}$$

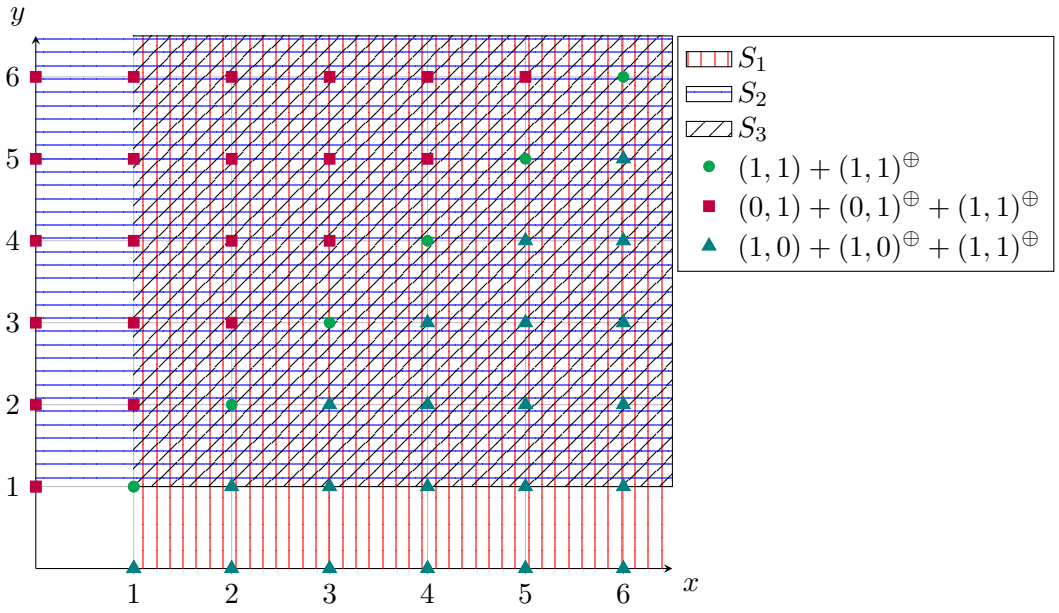

Figura 4.1: Representación original de S

$$\underline{S_1} = \frac{x}{(1-x)(1-y)} \qquad \underline{S_2} = \frac{y}{(1-x)(1-y)} \qquad \underline{S_3} = \frac{xy}{(1-x)(1-y)}$$
$$E_1 = (1,0) + \{(1,0),(0,1)\}^{\oplus} \qquad E_2 = (0,1) + \{(1,0),(0,1)\}^{\oplus} \qquad E_3 = (1,1) + \{(1,0),(0,1)\}^{\oplus}.$$

$$\begin{aligned} C &= (S_1 \cap S_2 \cap S_3) \cup (S_1 \cap {S_2}^c \cap {S_3}^c) \cup ({S_1}^c \cap S_2 \cap {S_3}^c) \\ &= (S_3) \cup (S_1 \cap {S_2}^c) \cup (S_1^c \cap S_2) \\ &= (1,1) + \{(1,0),(0,1)\}^{\oplus} \cup (1,0) + \{(1,0)\}^{\oplus} \cup (0,1) + \{(0,1)\}^{\oplus} \end{aligned}$$

$$\begin{aligned} \underline{S} &= \frac{xy}{(1-x)(1-y)} + \frac{x}{(1-x)} + \frac{y}{(1-y)} \\ &= \frac{xy}{(1-x)(1-y)} + \frac{x(1-y)}{(1-x)(1-y)} + \frac{y(1-x)}{(1-x)(1-y)} \\ &= \frac{xy + x(1-y) + y(1-x)}{(1-x)(1-y)}. \end{aligned}$$

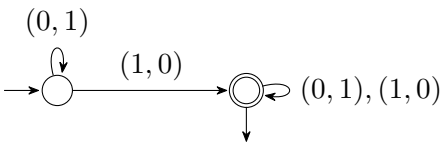

Figura 4.3: $\mathcal{A}_1$: Autómata de S_1

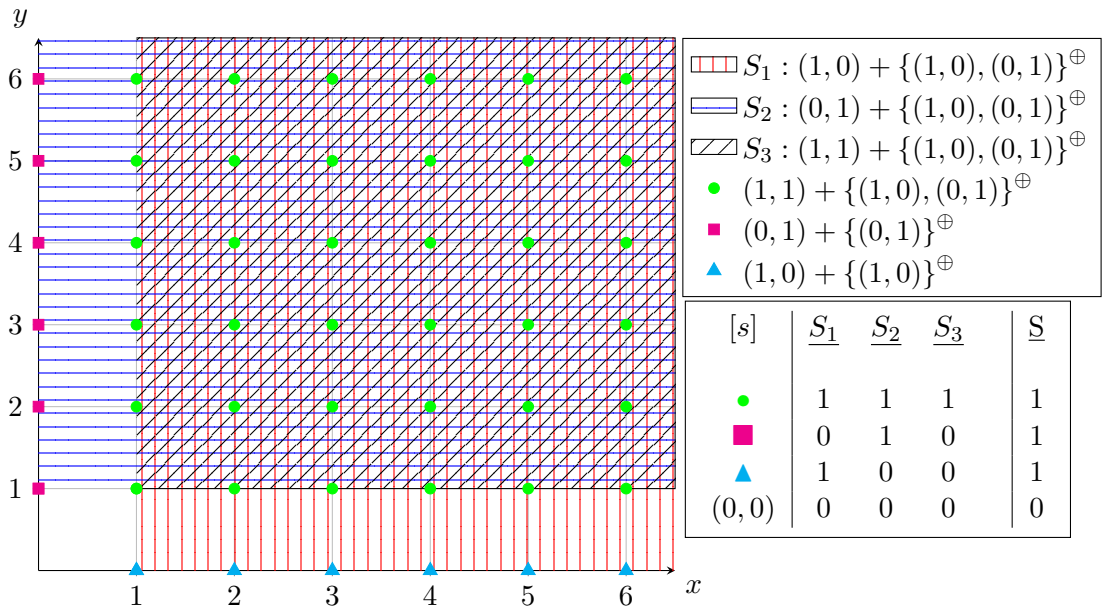

Figura 4.2: Representación alternativa de S utilizando las clases de equivalencia

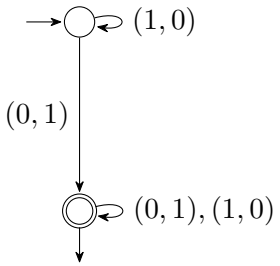

Figura 4.4: $\mathcal{A}_2$: Autómata de S_2

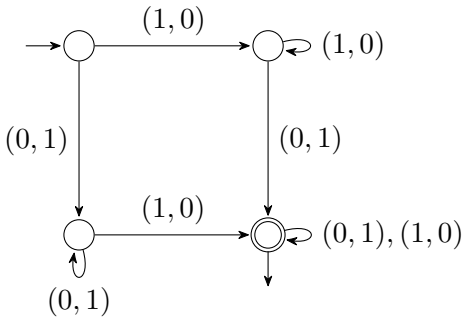

Figura 4.5: $\mathcal{A}_3$: Autómata de $S_3 \quad = S_1 \cap S_2 \cap S_3$

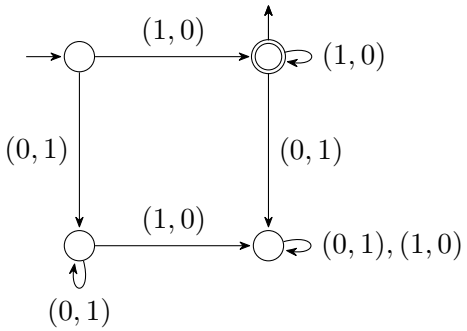

Figura 4.6: Autómata de $S_1 \cap {S_2}^c \cap {S_3}^c$

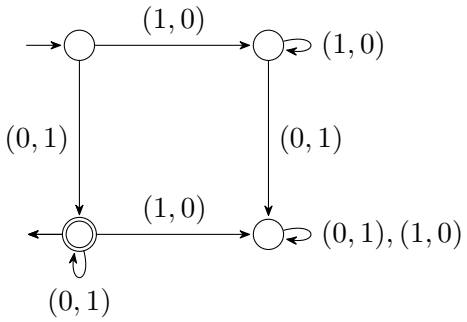

Figura 4.7: Autómata de ${S_1}^c \cap S_2 \cap {S_3}^c$

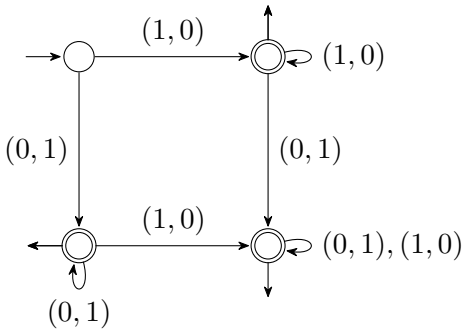

Figura 4.8: Autómata de $S = (S_1 \cap S_2 \cap S_3)^c \cap (S_1 \cap {S_2}^c \cap {S_3}^c)^c \cap ({S_1}^c \cap S_2 \cap {S_3}^c)^c$

Comentario. *El autómata que se obtiene de la demostración del Teorema 2 no siempre es mínimo. Por ejemplo, el de la Figura 4.8 no es el mínimo. El autómata mínimo se da a continuación y se puede obtener aplicando el algoritmo de Moore.*

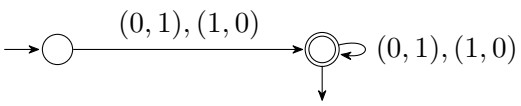

Figura 4.9: $\mathcal{A}_{\text{mín}}$: Autómata mínimo de S

5. ALGORITMO

En este capitulo probamos el Teorema 1 dando el siguiente algoritmo:

Input: E una expresión regular en A^*. Consideramos $S = \varphi(|E|) \in Rat(\mathbb{N}^k)$, la imagen conmutativa.

Output: Si el S dado en el input es reconocible, el output es un AFD $\mathcal{A} = \langle Q, A, E, I, T\rangle$ completo que acepta $\varphi^{-1}(S) = \mathcal{C}(|E|)$. Si no aviso.

1. Obtener una expresión semi-simple E' para S usando la Proposición 3
2. Hacer la expresión semi-simple E' consistente usando el Lema 17 (opcional).
3. Calcular a partir de E' la serie característica $\underline{S} = \dfrac{P_0}{Q_0}$, usando la Definición 2.13.
4. Factorizar P_0 y Q_0. Simplificar todos los factores posibles de $\dfrac{P_0}{Q_0}$ hasta obtener P y Q tales que $\dfrac{P}{Q}$ irreducible.
5. Si Q tiene algún factor de mas de una variable entonces no es reconocible. Cortamos y avisamos que no es regular $\mathcal{C}(|E|)$.
6. Obtener una expresión caracolada consistente C a partir de los polinomios P y Q usando el Teorema 3.
7. Obtener el autómata $\mathcal{A}$ usando Teorema 2 sobre la expresión C del paso anterior.

Comentario. *El paso 2 del algoritmo es opcional, luego del paso 4 siempre se obtienen los mismos polinomios porque estos son únicos para cualquier conjunto racional, ver Proposición 6. Sin embargo, se deja este paso por claridad y para simplificar el análisis de complejidad.*

5.1. Complejidad

Usaremos la notación asintótica usual O grande y diremos que para funciones $f, g : \mathbb{N} \to \mathbb{R}$, $f(n)$ es $\mathcal{O}(g(n))$ si existe C tal que para todo n suficientemente grande $|g(n)| \leq |Cf(n)|$.

Asumimos un alfabeto de k símbolos.

Consideramos el tamaño de una expresión racional a la cantidad de símbolos de la expresión. Por ejemplo, $a \cup b^*$ tiene tamaño 4. Siempre considerando expresiones escritas utilizando unicamente elementos de una base libre de generadores del monoide. Por ejemplo, una expresión para 2 en $\mathbb{N}$ tiene por lo menos tamaño 3 ya que la menor expresión racional que lo denota es $1 + 1$.

5.1.1. Complejidad de estados

Por complejidad de estados de un lenguaje regular nos referimos a la mínima cantidad de estados de un AFD completo que acepte ese lenguaje [12]. Es una medida natural para las operaciones de lenguajes regulares y a su vez nos da una cota inferior para la complejidad temporal y espacial de las operaciones entre autómatas. El estudio de la complejidad de estados se remonta por lo menos a [8].

Si bien el problema lo planteamos desde una expresión regular, la cota la daremos sobre la expresión semi-simple del conjunto en $\mathbb{N}^k$. Por lo que no estamos considerando el costo que implica desambiguar la expresión racional.

Comentario. *Notemos que el costo de desambiguar una expresión racional E que denota un conjunto en $\mathbb{N}$ es exponencial en la longitud de E. Dada la biyección entre $\mathbb{N}$ y a^*, desambiguar E en $\mathbb{N}$ es lo mismo que determinizar el autómata finito sobre el lenguaje a^* que surge de E. Y por el resultado de Marek [2] sabemos que cuando el alfabeto es unario, la cantidad de estados de un $AFND$ equivalente a un $AFND$ es, en el peor caso, exponencial en la cantidad de estados n del AFND dado, $\mathcal{O}\left(e^{\sqrt{n.log(n)}}\right)$.*

Sea una expresión semi-simple:

$$E = \bigcup_{i \in I} c_i + B_i^{\oplus}.$$

Para cada $j = 1, \dots k$, definimos α_j como el mínimo común múltiplo de las bases j-primarias de la expresión semi-simple, o α_j es 2 si no hay ninguna base j-primaria para la j-ésima componente, ya que necesitamos un estado trampa.

Proposición 19. *Para cada $j \in \{1, \dots, k\}$, $\alpha_j = \mathcal{O}\left(e^{\sqrt{n.log(n)}}\right)$, con n el tamaño de la expresión semi-simple E.*

Consultar [2] para más detalles sobre esta cota.

Proposición 20. *La complejidad de estados del autómata construido por nuestro algoritmo, tomando una expresión semi-simple E, es a lo sumo $\mathcal{O}\left(e^{k\sqrt{n.log(n)}}\right)$ con n la longitud de la expresión E.*

Demostración. Al introducir las expresiones caracoladas y sus autómatas observamos que la máxima cantidad de estados por dimensión no aumenta al realizar operaciones booleanas entre autómatas provenientes de expresiones caracoladas consistentes (Proposición 16). Por lo tanto, partiendo de una expresión semi-simple consistente

$$E' = \bigcup_{i \in I'} c_i' + B_i^{\oplus}$$

y usando la Proposición 12, la cantidad de estados por dimensión se puede acotar por

$$l_j = \max_{i \in I'}\{|c_i'|_j\} + \alpha_j - 1$$

Luego, la complejidad de estados de un autómata proveniente de una expresión semi-simple E' consistente es a lo sumo

$$\prod_{j=1}^{k} l_j = \prod_{j=1}^{k} (\max_{i \in I}\{|c_i'|_j\} + \alpha_j - 1) = \mathcal{O}\big((c_{max} + \alpha_{max})^k\big),$$

donde
$\alpha_{max} = \max_{j\in\{1,\dots,k\}} \alpha_j = \mathcal{O}\Big(e^{\sqrt{n.log(n)}}\Big)$,
$c_{max} = \max_{i\in I} |c_i| = \mathcal{O}(n)$.

Dado que c_{max} se corresponde con la parte finita de E, c_{max} se puede acotar por el tamaño de la expresión.

Para transformar la expresión semi-simple E en una consistente E' necesitamos cambiar cada una de las bases j-primarias por $\alpha_j\ e_j$. En el peor caso, por cada dimensión, debemos sumarle a algún término c hasta $(\alpha_j - 1)e_j$. Luego

$$\max_{i\in I}\{|c'_i|_j\} \leq \max_{i\in I}\{|c_i|_j\} + (\alpha_j - 1).$$

Esto no altera la complejidad de estados total,

$$\begin{aligned}\prod_{j=1}^{k}(\max_{i\in I}\{|c_i|_j\} + 2(\alpha_j - 1)) &= \mathcal{O}((c_{max} + \alpha_{max})^k)\\ &= \mathcal{O}\bigg(\Big(n + e^{\sqrt{n.log(n)}}\Big)^k\bigg)\\ &= \mathcal{O}\bigg(\Big(e^{\sqrt{n.log(n)}}\Big)^k\bigg).\end{aligned}$$

□

Comentario. *No obstante, si elegimos reducir al máximo los polinomios llegaremos al menor α_j posible por lo que en general obtenemos un autómata lo más chico posible en cada una de sus dimensiones. Sin embargo, como ya pudimos ver en el Ejemplo 4.1, por más que el algoritmo comience con P/Q irreducibles, el autómata resultante no necesariamente es el autómata mínimo.*

Comentario. *Hoffmann en [6] prueba para el caso de los lenguajes de grupos una complejidad de estados de $\mathcal{O}(n^k e^{k\sqrt{n.log(n)}})$, con n la cantidad de estados del autómata de permutación.*

5.1.2. Complejidad temporal

Llamamos operaciones elementales a cualquier operación aritmética sobre números naturales, racionales o reales.

Proposición 21. *Nuestro algoritmo tiene una complejidad temporal de $\mathcal{O}(m^2 2^m + m^{7k})$ operaciones elementales en el peor caso, donde $m = \mathcal{O}\Big(e^{k\sqrt{n.log(n)}}\Big)$, n es el tamaño de la expresión semi-simple E y k es el tamaño del alfabeto.*

Demostración. Para transformar la expresión semi-simple E en una consistente E' necesitamos cambiar todas las bases j-primarias por $\alpha_j\ e_j$. Para eso necesitamos considerar a lo sumo $n\alpha_1 \cdots \alpha_k$ términos simples que usaremos para construir la expresión E' conforme al Lema 17.

Luego E' constará de $t = \mathcal{O}\Big(|I| \prod_{j=1}^{k} \alpha_j\Big)$ términos simples. Dado que $|I|$ es una unión disjunta y asumimos un alfabeto fijo, podemos acotar $\mathcal{O}((c_{max} + \alpha_{max})^k)$, sino tendríamos términos repetidos. Ademas, $\mathcal{O}(\prod_{j=1}^{k} \alpha_j) = \mathcal{O}\big({\alpha_{max}}^k\big)$. Luego,

$$t = \mathcal{O}\Big(((c_{max} + \alpha_{max})\alpha_{max})^k\Big) = \mathcal{O}(m^2).$$

Al pasar a serie, cada término de la expresión semi-simple consistente genera una fracción. Al tomar denominador común, en el peor caso, tendremos que multiplicar a cada término por el denominador de los restantes. Notemos que cada denominador tiene a lo sumo k factores de la forma $(1 - \underline{d})$, pues la base de la que provienen tiene a lo sumo k elementos.

Si distribuimos cada uno de los denominadores, el polinomio tendrá a lo sumo 2^k términos. Entonces, al tomar denominador común, tendremos que multiplicar cada numerador por los $(t-1)$ denominadores restantes. Por lo tanto, quedan $2^k(t-1)t$ términos en el numerador. Notar que $\mathcal{O}\Big(2^k t^2\Big) = \mathcal{O}(t^2)$.

Ahora tenemos que simplificar los factores del denominador de más de una variable. Sin embargo, por una cuestión de simplicidad optamos por reducir al máximo P_0/Q_0. Para eso factorizamos P_0 y Q_0 con costo $\mathcal{O}(m^{7k}), ver[7]$. Esta cota se debe a que podemos acotar el grado de cada variable del polinomio por $\mathcal{O}(c_{max} + \alpha_{max})$ y los coeficientes como son de una serie característica son 1 o -1.

Luego de factorizar y reducir no puede haber mas de m términos en el caso reconocible pues las expresiones caracolada tienen todas la misma parte infinita y no puede haber mas de $(c_{max} + \alpha_{max})^k$ por el mismo argumento que dimos para acotar $|I|$, solo que acá la parte infinita sabemos que debe coincidir. Por lo tanto tendremos a lo sumo m autómatas a interesecar. Como ya se vio la cantidad máxima de estados es $\mathcal{O}(m)$. Además como Y, el conjunto que definimos en la demostración del Teorema 3, son soluciones de un sistema de ecuaciones lineales de m variables binarias, $|Y| = \mathcal{O}(2^m)$. Por lo tanto tendríamos $\mathcal{O}(m2^m)$ operaciones booleanas, cada una con costo lineal en m. Luego, el costo total de computar el autómata final a partir de los polinomios es de $\mathcal{O}(m^2 2^m)$. Obtenemos una complejidad temporal total de peor caso $\mathcal{O}(m^2 2^m + m^{7k} + m^4)$ que es

$$\mathcal{O}(m^2 2^m + m^{7k}).$$

□

Bibliografía

[1] Cano, A., Guaiana, G., and Éric Pin, J. Regular languages and partial commutations. Information and Computation 230 (2013), 76–96.

[2] Chrobak, M. Finite automata and unary languages. Theor. Comput. Sci. 47 (1986), 149–158.

[3] Eilenberg, S., and Schützenberger, M. Rational sets in commutative monoids. Journal of Algebra 13, 2 (1969), 173–191.

[4] Ginsburg, S., and Spanier, E. H. Bounded regular sets. Proceedings of the American Mathematical Society 17, 5 (1966), 1043–1049.

[5] Gohon, P. An algorithm to decide whether a rational subset of $\mathbb{N}^k$ is recognizable. Theoretical Computer Science 41 (1985), 51–59.

[6] Hoffmann, S. State complexity bounds for the commutative closure of group languages. J. Autom. Lang. Comb. 28, 1-3 (2023), 27–57.

[7] Lenstra, A. K. Factoring multivariate integral polynomials. Theor. Comput. Sci. 34 (1984), 207–213.

[8] Maslov, A. N. Estimates of the number of states of finite automata. Sov. Math., Dokl. 11 (1970), 1373–1375.

[9] Parikh, R. J. On context-free languages. J. Assoc. Comput. Mach. 13 (1966), 570–581.

[10] Sakarovitch, J. Elements of automata theory. Cambridge University Press, Cambridge, 2009. Translated from the 2003 French original by Reuben Thomas.

[11] Thierrin, G. Permutation automata. Math. Syst. Theory 2 (1968), 83–90.

[12] Yu, S. Regular Languages. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 41–110.