Autómatas finitos y lenguajes formales

Trabajo fin de grado

Curso 2014-15

David Navarro Fernández

20 de julio de 2015

Índice general

Introduction

When a matematician wants to solve a mathematical problem, he can try to solve it with the tools he has learned or he can use a computer program. Different computer programs have been developed as a help to solve different mathematical problems. However there are problems that can not be solved using computers. Turin's Machine is a method we can use to study the decidability of a problem. But we can be more interested in that moment when we introduce all the problem's data in the computer. It's usual to make spelling mistakes or choose the wrong methods or operations. It is known that the computer works with ones and zeros, but we use symbols, words, numbers. The computer has to convert all that stuff into its own language but that has to be done following the rules of the language and checking that everything is correct. That's the first step: compilation.

This is the motivation of our work, although we are only going to analize the mathematical structure of finite automata that are used in the first step of the compilation, that is, the lexical analysis.

Our work belongs to the field of knowledge of algebra and computer science, in particular semigroups and finite automata. Semigroups have simpler structure than groups, only the associative property is required. However that makes their study more difficult. Concerning automata, there are different kinds of them, but we're going only to study finite automata.

An automaton has an abstract structure that can be represented by a graph and can recognize a certain formal language. A formal language is a subset of the free monoid with basis a finite alphabet. For example, suppose we have an automaton which recognize all strings of the English language. Then, we can prove that the string "pride" is recognized by the automaton, however "stal" isn't. There are different kinds of formal languages. Each of them can be described by a set of well defined formulae, that is, by a grammar. According to Chomsky hierarchy, we can represent with a table all kinds of languages, automata and grammars.

Clasificación	**Lenguages**	**Grammar**	**Automata**	**Difficulty**
Type 0	recursively enumerable	without restrictionss	Turing machine	undecidable
Type 1	context-sensitive	context-sensitive	linear bounded automata	exponential
Type 2	context-free	context-free	pushdown automata	polynomic
Type 3	regular	regular	finite automata	linear

Our work will be devoted to automata of type 3: the finite automata. They are the simplest automata. An example will help us to understand its performance. Imagine a room with the light off and one switch. Figure 1 represents this situation. Suppose the light is initially off, this is the initial state. Each time we push the switch, the state changes. If we push one time, the light changes to "on". If we push twice the light changes to "off". In this automaton "on" and "off" are the states of the automaton. The arrows that go from one state to another are named transitions. Automata are designed to recognize certain types of languages and the states that recognize them are called final states (usually represented by a double circle). In our automaton the two states are final and '"push" would be unique letter in the alphabet.

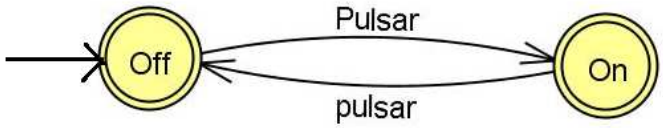


Figura 1: Light switch in a room

At the begining, finite automata were created by McCulloch and Pitt as a mathematical model for neuronal connections in the brain. Although their model was not correct, Stephen Kleene in 1956 used some structures (Pitt's models) that recognized certain languages. During that time the basic concepts and theorems about this theory were setteld down. The researchers involved were among others Nerode, Moore, Chomsky, Mealy, Myhill, Shützenberger, etc. A part of this theory is developed in this work. Among the applications of finite automata we have the following

1. Software to design and test the perfomance of integrated circuits such as microprocessors.
2. Lexical analyzer of a regular compiler.
3. Software to explore long texts.

4. Software to check procedures and systems of different kinds with a finite number of states such that comunication's protocols or protocols to exchange information in a safe way.

Finally I will describe the contents of this work more specifically. In the first chapter we introduce the basic definitions and results on semigroups and automata. In chapter two we define regular languages as those who are accepted by an automaton. Then, Myhill-Nerode theorem for semigroups (theorem 2.2.9) is proved and we show that regular languages are precisely those sets that are recognizable in the finitely generated free semigroup Σ^* with Σ a finite alphabet (theorem 2.2.11). The minimal automaton that recognizes a particular language is also considered. Finally, in the last chapter we define the languages given by rational expressions, the non-deterministic finite automata and systems of linear equations with formal languages. With all this we can prove Kleene's theorem (3.2.12), which together with previous results, leads to Corollary 3.2.13: regular languages coincide with those languages recognizable by the semigroup Σ^* and coincide with those given by rational expressions.

Introducción

Un matemático, cuando se enfrenta al cálculo de un problema, puede intentar resolverlo con las herramientas aprendidas o puede utilizar un programa de ordenador que se lo resuelva. Para cada tipo de problemas se han desarrollado diferentes programas que facilitan su resolución. Aunque no todos los problemas se pueden resolver utilizando los ordenadores, una forma de estudiar la decidibilidad de un problema es con las Máquinas de Turing. Pero podemos estar interesados en el paso previo: el de la introducción de datos en el ordenador. En este caso podemos tener errores en la escritura o podemos haber elegido los métodos u operaciones de una forma correcta o incorrecta. Sabemos que el ordenador trabaja con unos y ceros, y nosotros hemos introducido palabras, números, signos de puntuación, etc. Además de convertir lo escrito de forma correcta en su lenguaje, tiene que hacerlo conservando las reglas de estructura que había en el programa y comprobando que todo está correctamente escrito según las reglas del programa. Este es el primer paso que realiza nuestro ordenador: la compilación.

Esta es la motivación de este trabajo, aunque solo procederemos a analizar matemáticamente la estructura de los autómatas finitos que se utilizan en el primer paso de la compilación, el análisis léxico.

Nuestro trabajo se sitúa dentro del campo de conocimiento del Álgebra y de las Ciencias de la Computación, en concreto, semigrupos y autómatas finitos. El semigrupo es una estructura más simple que la de grupo pues solo cumple la propiedad asociativa, aunque eso dificulta su manejo. En relación a los autómatas hay diferentes tipos de autómatas, pero nosotros vamos a estudiar solo los autómatas finitos.

Un autómata es una estructura abstracta que se puede representar mediante un grafo y que tiene el objetivo de reconocer un determinado lenguaje formal. Un lenguaje formal es un subconjunto del monoide libre con base un alfabeto finito. Por ejemplo, supongamos que tenemos un autómata que reconoce todas las palabras del lenguaje español, entonces podemos comprobar que la palabra "melocotón" pertenece al lenguaje, en cambio "rrristo" no pertenece. Hay diversos tipos de lenguajes formales. Cada tipo puede ser definido por un conjunto de formulas bien definidas, es decir, puede ser definido por una gramática. Podemos representar los diferentes tipos de lenguajes, autómatas y gramáticas en una tabla, según la jerarquía de Chomsky.

Clasificación	Lenguajes	Grámatica	Autómata	Dificultad
Tipo 0	Recursivamente numerables	Sin restricciones	Maquina de Turing	Indecidible
Tipo 1	Dependiente del contexto	Dependiente de contexto	Autómata linealmente acotado	Exponencial
Tipo 2	Libres de contexto	Libre de contexto	Autómata con pila	Polinómico
Tipo 3	Regular	Regular	Autómata finito	Lineal

Nuestro trabajo se centra en los autómatas de tipo 3: los autómatas finitos. Son los autómatas más simples. Vamos a poner un ejemplo para entender su funcionamiento. Imaginemos una habitación con la luz apagada y con un interruptor. La figura 2 muestra un autómata que representa esta situación. Suponemos que inicialmente la luz está apagada, este es el estado inicial. Cada vez que pulsamos el interruptor cambiamos el estado. Si pulsamos una vez la luz pasa a "on", si son 2 veces la luz vuelve al estado "off". "on" y "off" son los estados del autómata. Las flechas que van de un estado a otro se llaman transiciones. Los autómatas son diseñados para reconocer un tipo de lenguaje y los estados que los reconocen son los estados finales (representados con dos círculos). En el ejemplo "pulsar" sería la única letra del alfabeto.

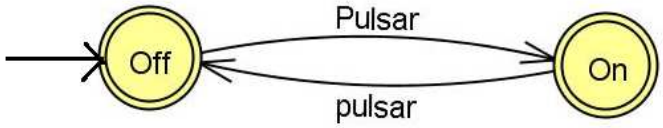


Figura 2: El interruptor de una habitación

Los autómatas finitos nacieron en un primer momento de la mano de McCulloch y Pitt con el objetivo de modelar las conexiones neuronales en el cerebro. Aunque se demostró que su modelo era erróneo, Stephen Kleene definió en 1956 unas estructuras (los modelos de Pitt) que podían reconocer unos lenguajes determinados. Durante esta década se obtuvo una serie de resultados que permitieron establecer los conceptos y teoremas básicos de la teoría de autómatas. Entre los investigadores destacados se encuentran Nerode, Moore, Chomsky, Mealy, Myhill, Shützenberger, etc. Alguna parte de esta teoría la desarrollaremos en este trabajo. Entre las utilidades de los autómatas finitos tenemos:

1. Software para diseñar y probar el comportamiento de circuitos digitales como los microprocesadores.

2. El analizador léxico de un compilador típico.

3. Software para explorar textos largos.

4. Software para verificar sistemas de todo tipo que tengan un número finito de estados diferentes, tales como protocolos de comunicaciones o protocolos para el intercambio seguro de información.

Finalmente voy a describir el contenido del trabajo de forma más concreta. En el primer capítulo se introducen las definiciones y resultados básicos de semigrupos y autómatas. En el segundo capítulo se definen los lenguajes regulares como aquellos que son aceptados por un autómata finito. Después se demuestra el teorema de Myhill-Nerode (teorema 2.2.9) para semigrupos y se prueba que los lenguajes regulares son precisamente los conjuntos reconocibles en el semigrupo finito-generado libre Σ^* con Σ un alfabeto finito (teorema 2.2.11). También se considera el autómata minimal que reconoce un determinado lenguaje. Finalmente, en el último capítulo se definen los lenguajes dados por expresiones racionales, los autómatas no deterministas y se introducen los sistemas de ecuaciones con lenguajes formales. Todo ello permite demostrar el teorema de Kleene (3.2.12), que junto con los resultados anteriores lleva al corolario 3.2.13: los lenguajes regulares coinciden con los reconocibles por el semigrupo Σ^* y coinciden con los dados por expresiones racionales.

Capítulo 1

Semigrupos y Autómatas

En este capítulo se introducen las definiciones y resultados básicos se semigrupos y autómatas. Las referencias principales son [1] para todo lo concerniente a autómatas y parte de lo relativo a semigrupos. Los restantes resultados de semigrupos se pueden encontrar en [2].

1.1. Semigrupos

Empezaremos introduciendo los semigrupos, una estructura muy importante del Álgebra, que nos será muy útil para el resto del trabajo.

Un semigrupo es un conjunto con una operación a la que solo se le exige que cumpla la propiedad asociativa. La definición, como se puede ver, es muy simple, aunque en su propia simplicidad está su complejidad.

Como se verá a lo largo del capitulo, las propiedades de los semigrupos se parecen a algunas propiedades de los grupos. Un grupo, en particular, es un semigrupo y, por tanto, hereda las propiedades de este último.

Definición 1.1.1. *Un* ***semigrupo*** *es un conjunto no vacío S junto con una operación binaria asociativa. Un* ***monoide*** *es un semigrupo con un elemento neutro.*

Es fácil de ver que un elemento neutro, si existe, es único.

Un semigrupo S que no es monoide se puede convertir en un monoide añadiéndole un elemento neutro como indica la siguiente definición

Definición 1.1.2. *Sea S un semigrupo y sea 1 un elemento distinto de todos los de S. Si S es un monoide se define $S^1 = S$. Si S no es un monoide se define el semigrupo $S^1 = S \cup \{1\}$ con la misma operación de S y $x1 = 1x = x$ para todo $x \in S$. Entonces, S^1 es un monoide.*

Definición 1.1.3. *Sea S un semigrupo. Un elemento $c \in S$ se llama un* ***cero*** *de S si cumple $cs = sc = c$ para todo s de S. Un elemento e es* ***idempotente*** *cuando $e^2 = e$.*

Es fácil de ver que un cero, si existe, es único. En este caso, se denotará por 0.

Un semigrupo S sin cero se puede convertir en un semigrupo con cero añadiéndole un elemento 0 distinto de los de S de forma análoga a como se ha hecho en la definición 1.1.2.

Definición 1.1.4. *Un **subsemigrupo** de un semigrupo S es un subconjunto no vacío de S cerrado para la operación de S. Si S es monoide, un **submonoide** es un subsemigrupo que contiene el elemento neutro.*

Ejemplo 1.1.5. Sea Q un conjunto y $\mathcal{R}(Q)$ el conjunto de las relaciones binarias sobre Q, es decir, el conjunto de todos los subconjuntos de $Q \times Q$. Si ρ es una relación binaria y $(a, c) \in \rho$, se suele escribir $a\ \rho\ c$ y se lee "*a relacionado con c*". Dadas dos relaciones $\tau, \rho \in \mathcal{R}(Q)$, definimos la relación $\tau \cdot \rho$ por

$$a\ (\tau \cdot \rho)\ b \Leftrightarrow \exists\ c \in Q\ \mid\ a\ \rho\ c\ \text{ y }\ c\ \tau\ b.$$

Se obtiene de esta forma un semigrupo llamado **semigrupo de las relaciones de** Q. La operación definida se llama **producto relacional**.

Definimos ahora las aplicaciones (o funciones), que son elementos de $\mathcal{R}(Q)$.

Definición 1.1.6. *Una **aplicación** f de Q en Q, escrito $f : Q \to Q$, es un elemento de $\mathcal{R}(Q)$ que cumple que para todo $q \in Q$, existe un único $p \in Q$ tal que $(q, p) \in f$. En este caso, se escribe $f(q) = p$ y al producto relacional de aplicaciones se le llama* **composición de aplicaciones***, que es fácil ver que es a su vez otra aplicación. Denotaremos por $F_l(Q)$ al monoide de las aplicaciones de Q en Q con la composición de aplicaciones, que se denotará con los operadores actuando a la derecha, es decir, $(f \circ g)(a) = f[g(a)]$. Entonces $F_l(Q)$ es un subsemigrupo de $\mathcal{R}(Q)$.*

Al monoide opuesto $F_l(Q)$[1] lo denotaremos por $F_r(Q)$. Si la operación en $F_r(Q)$ se denota también por $\circ$, escribiremos los operadores actuando a la derecha, es decir, $(a)(f \circ g) = [(a)f]g$.

Podemos también considerar las aplicaciones de Q en Q definidas parcialmente, es decir, cuyo dominio es un subconjunto de Q. A estos monoides se les denotará por $PF_l(Q)$ y $PF_r(Q)$ y son monoides con cero (la aplicación vacía).

El conjunto de funciones $F_r(Q)$ será usado para definir las transiciones de los autómatas y el conjunto $PF_r(Q)$ definirá las transiciones de los autómatas cuando estos no sean completos.

Dado un elemento $q \in Q$, sea c_q la aplicación constante $c_q(x) = q$ para todo $x \in Q$. Observemos que un subconjunto de $PF_r(Q)$ (o $PF_l(Q)$) formado exclusivamente por funciones constantes es un subsemigrupo que cumple $xy = y$ para todo x, y. Se puede ver fácilmente que todo elemento es idempotente.

[1]El **opuesto** de un semigrupo S se define como el conjunto S con la operación $x \cdot y = yx$.

Ejemplo 1.1.7. Si S es un semigrupo y T es un conjunto, el conjunto S^T de las aplicaciones de T en S tiene estructura de semigrupo con la operación $(fg)(t) = f(t)g(t)$. Si S es un monoide también lo es S^T. S se puede ver como un subsemigrupo de S^S identificando cada elemento $s \in S$ con la función constante $c_s \in S^S$.

Ejemplo 1.1.8. Dados un conjunto $\Sigma = \{a\}$, definimos

$$\Sigma^+ = \{a, a^2, a^3, \ldots, a^n, \ldots\}$$

con la operación obvia. Es el semigrupo libre con base Σ (ver la definiciíon 1.1.22).

Ejemplo 1.1.9. Dados dos enteros positivos r y m, el conjunto

$$C_{m,r} = \{a, a^2, \ldots, a^r, \ldots, a^{r+m-1}\}$$

con $a^{r+m} = a^r$ es un semigrupo. Es fácil ver que $\{a^r, \ldots, a^{r+m-1}\}$ es un subgrupo cíclico de $C_{m,r}$.

Ejemplo 1.1.10. En M_n, el semigrupo multiplicativo del anillo de las matrices $n \times n$ sobre $\mathbb{Z}/2\mathbb{Z}$, consideremos las matrices con a lo sumo una entrada distinta de cero. Este conjunto es un subsemigrupo de M_n que se suele denotar por B_n.

Ejemplo 1.1.11. Sea S un semigrupo. Si A y B son subconjuntos de S se define $A \cdot B$ como

$$A \cdot B = \{ab \mid a \in A, \quad b \in B\}.$$

Con esta operación así definida, el conjunto de las partes de S, $\mathcal{P}(S)$, tiene estructura de semigrupo con elemento cero (el conjunto vacío). Se puede ver que si S es un grupo, un elemento distinto de cero es un idempotente si y solo si es un subgrupo de G.

Definición 1.1.12. *Un* **homomorfismo** *entre semigrupos S y T es una aplicación $f : S \to T$ que cumple $f(ab) = f(a)f(b)$ para todo $a, b \in S$. Si, además, S y T son monoides y se cumple $f(1) = 1$ entonces f se llamará un* **homomorfismo de monoides**. *Se define monomorfismo, epimorfismo e isomorfismo de la forma habitual.*

Definición 1.1.13. *Dado un semigrupo S y un elemento $s \in S$, denotemos por δ_s la* **multiplicación a la derecha** *por s, es decir, $\delta_s(x) = xs$. Entonces tenemos un homomorfismo de semigrupos:*

$$\begin{array}{rccc} \hat{\delta} & : & S & \longrightarrow & F_r(S) \\ & & s & \mapsto & \delta_s \end{array}$$

llamado **representación regular a la derecha** *de S. Si S es un monoide, entonces la representación es* **fiel**, *es decir, $\hat{\delta}$ es inyectiva.*

Definición 1.1.14. *Dado un semigrupo S, una relación de equivalencia θ en S se llama* **invariante por la derecha** *si para todo $u, s, t \in S$ se cumple que $s\ \theta\ t$ implica su $\theta\ tu$. De forma análoga definimos relación de equivalencia* **invariante por la izquierda**. *Una* ***congruencia*** *en S es una relación de equivalencia θ invariante por la derecha y por la izquierda.*

Todo semigrupo tiene dos congruencias que llamaremos triviales: la identidad, que denotaremos por id en la que cada elemento solo está relacionado consigo mismo, y aquella en que cada elemento está relacionado con todos los demás.

Se puede ver que una relación de equivalencia θ en un semigrupo S es una congruencia si y solo si se cumple que $\forall s,t,u,v \in S$, $s\ \theta\ t$ y $u\ \theta\ v$ implica $su\ \theta\ tv$.

Proposición 1.1.15. *Sea θ una congruencia en un semigrupo S. Entonces, el conjunto cociente S/θ tiene estructura de semigrupo con la operación $[s]_\theta[t]_\theta = [st]_\theta$, donde $[s]_\theta$ denota la clase de equivalencia del elemento s.*

Demostración. [1, página 202]. ■

Definición 1.1.16. *Dada una congruencia θ en un semigrupo S, se define el* **índice** *de θ como el cardinal del conjunto cociente S/θ.*

Proposición 1.1.17. *Sean θ_1 y θ_2 son congruencias en un semigrupo S de índices finitos m_1 y m_2 respectivamente. Entonces, $\theta_1 \cap \theta_2$ es una congruencia de índice menor o igual que $m_1 m_2$.*

Demostración. Se tiene una aplicación inyectiva

$$S/\theta_1 \cap \theta_2 \to S/\theta_1 \times S/\theta_2$$

de donde el resultado sigue. ■

Definición 1.1.18. *Sea S un conjunto, θ una relación de equivalencia sobre S y $X \subseteq S$. Decimos que X es* ***saturado por*** *θ o que θ* **satura a** *X si la siguiente condición se satisface:*

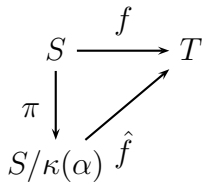
$$(x \in X, \quad y\,\theta\,x) \to y \in X.$$

Equivalentemente, X es saturado por θ si y solo si X es unión de clases de equivalencia, es decir,

$$X = \bigcup_{x \in X} [x]_\theta.$$

Dado un homomorfismo $f : S \to T$ podemos definir en S la congruencia **núcleo** de f definida por $x\ \kappa(f)\ y \Leftrightarrow f(x) = f(y)$.

Teorema 1.1.19. *Sea $f : S \to T$ un homomorfismo de semigrupos. Sea $\pi : S \to S/\kappa(f)$ el paso al cociente. Entonces hay una única aplicación $\hat{f} : S/\kappa(f) \to T$ inyectiva tal que $\hat{f}\pi = f$. La aplicación $\hat{f}$ da lugar a un isomorfismo entre $S/\kappa(f)$ y $f(S)$.*



La función $\hat{f}$ es sobreyectiva si y solo si f es sobreyectiva.

DEMOSTRACIÓN. [1, página 203]. ■

Definición 1.1.20. *Una congruencia θ se dice que es* **más fina** *que otra ρ si $\theta \subseteq \rho$, es decir, si $s\ \theta\ t$ implica $s\ \rho\ t$.*

Proposición 1.1.21. *Sea θ es una congruencia más fina que otra ρ. Si θ es de índice finito, entonces también lo es ρ y además el índice de ρ es menor o igual que el índice de θ.*

DEMOSTRACIÓN. Se tiene un epimorfismo $S/\theta \to S/\rho$, de donde el resultado sigue ([1, página 17]). ■

Definición 1.1.22. *Sea Σ un conjunto no vacío. El* **semigrupo libre** *con base Σ, que denotaremos por Σ^+, es el conjunto de las sucesiones finitas de elementos de Σ con la operación de concatenación.*

Es obvio que Σ^+ cumple la siguiente propiedad universal: toda aplicación de Σ a un semigrupo S puede extenderse de forma única a un homomorfismo de Σ^+ en S.

En el contexto de los lenguajes formales, Σ suele ser un conjunto finito llamado *alfabeto* y los elementos de Σ^+ se llaman *palabras*. De hecho, se definirá un *lenguaje formal en el alfabeto* Σ a cualquier subconjunto de Σ^+.

Podemos añadir a Σ^+ la palabra vacía o neutra, que denotaremos por ϵ, y obtener así un monoide Σ^*, que es el monoide libre sobre Σ.

1.2. Autómatas finitos

Vamos a empezar esta sección definiendo una estructura básica de este trabajo: el autómata finito. En esta parte solamente nos interesamos por las propiedades del autómata como estructura. No hay estados iniciales ni finales y, por lo tanto, los autómatas no reconocen lenguajes. Obtendremos una estructura a partir de las transiciones del autómata que serán interpretadas como elementos de $F_r(Q)$ (ver definición 1.1.6) y darán lugar a un semigrupo.

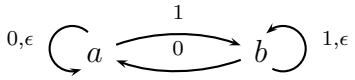
Definición 1.2.1. *Un* **autómata finito**, *que abreviaremos por AF, es una terna $A = (Q, \Sigma, \delta)$ donde Q y Σ son conjuntos finitos no vacíos y δ es una aplicación*

$$\delta : Q \times \Sigma \longrightarrow Q$$

A Q se le llama el **conjunto de estados** *y a Σ el* **alfabeto**. *A la función δ se la llama a veces* **función de transición**. *Dado un estado q y una letra $a \in \Sigma$, normalmente denotaremos por qa a la imagen de (q, a) por δ.*

En muchos textos, la aplicación δ no está definida en todo $Q \times \Sigma$ (ver la definición 3.1.1), lo que resulta útil para el diseño de autómatas reconocedores de lenguajes. En este caso, nuestra definición correspondería a un **autómata completo o determinado**.

Un AF se representa muchas veces mediante un grafo dirigido. Por ejemplo, el **flip-flop** viene dado por $Q = \{a, b\}$, $\Sigma = \{0, 1, \epsilon\}$ y $a \cdot \epsilon = a$, $b \cdot \epsilon = b$, $a \cdot 0 = a$, $a \cdot 1 = b$, $b \cdot 0 = a$, $b \cdot 1 = b$ y su grafo es:



Dada una palabra $w = a_1 a_2 \cdots a_n \in \Sigma^+$ con $a_i \in \Sigma$ y un estado inicial q, el autómata evoluciona pasando a través de una sucesión de estados:

$$q, qa_1, qa_1a_2, \ldots, qa_1a_2 \cdots a_n.$$

Por lo tanto, podemos extender la aplicación δ a $Q \times \Sigma^+$ definiendo $\delta(q, a_1a_2 \cdots a_n) = qa_1a_2 \cdots a_n$. También podemos extender δ a $Q \times \Sigma^*$ añadiendo $\delta(q, \epsilon) = q$.

La existencia y unicidad de esta extensión, que como ya se ha indicado denotaremos por la misma δ, viene dada formalmente por la siguiente proposición.

Proposición 1.2.2. *$A = (Q, \Sigma, \delta)$ un AF. Entonces hay una única función $\Delta : Q \times \Sigma^* \to Q$ que satisface las siguientes condiciones:*

(i) $\Delta(q, \epsilon) = q$, $q \in Q$.

(ii) $\Delta(q, a) = \delta(q, a)$, $q \in Q$, $a \in \Sigma$.

(iii) $\Delta(q, aw) = \Delta(\delta(q, a), w)$, $q \in Q$, $a \in \Sigma$, $w \in \Sigma^*$.

Demostración. [1, pagina 17]. ∎

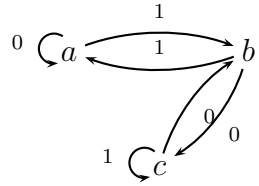
A través de la extensión de δ, el semigrupo Σ^+ actúa sobre el conjunto Q, pues cada elemento $w \in \Sigma^+$ define una aplicación

$$\delta_w : Q \longrightarrow Q$$

dada por $\delta_w(q) = qw$. Tenemos entonces un homomorfismo de semigrupos

$$\begin{array}{rccc} \hat{\delta} & : & \Sigma^+ & \longrightarrow & F_r(Q) \\ & & w & \mapsto & \delta_w \end{array}$$

Ejemplo 1.2.3. *Un ejemplo sencillo e interesante de AF $A = (Q, \Sigma, \delta)$ es el siguiente:*



El conjunto de estados es $Q = \{0, 1, 2\}$, y el alfabeto es $\Sigma = \{0, 1\}$. Dado un número en base 2, con este autómata podemos conocer cual es el resto en módulo 3.

Definición 1.2.4. *Dado un AF $A = (Q, \Sigma, \delta)$, se define el* **semigrupo asociado a** *A, denotado por $S(A)$, como la imagen de $\hat{\delta}$, es decir, $S(A)$ es el subsemigrupo de $F_r(Q)$ generado por las aplicaciones δ_a con $a \in \Sigma$.*

La aplicación $\hat{\delta}$ puede también extenderse a Σ^* definiendo δ_ϵ como la identidad de Q. Se define el **monoide asociado a** *A, denotado por $M(A)$, como la imagen de esta extensión. Podeis encontrar definido en algunos libros como* **monoide de transición de** *A.*

Según el teorema 1.1.19 tenemos que

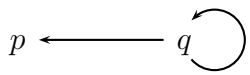
$$S(A) = \Sigma^+/\kappa(\hat{\delta})$$

lo que nos da otra forma de ver el semigrupo del autómata: es el semigrupo cociente de Σ^+ por la relación

$$w \sim w' \Leftrightarrow \delta_w = \delta_{w'}.$$

Por otra parte, $S(A)$ puede tener elemento neutro y ser, por tanto, un monoide pero no tiene porque coincidir con $M(A)$. De hecho, $M(A)$ se obtiene a partir de $S(A)$ añadiéndole la aplicación identidad de Q.

Ejemplo 1.2.5. Consideremos el autómata con $Q = \{p, q\}$ y alfabeto $\Sigma = \{a\}$ cuyo grafo es



Entonces, δ_a es la función constante y $S(A) = \{\delta_a\}$, es decir, $S(A)$ es el semigrupo trivial, que evidentemente es un monoide. Sin embargo, $M(A) = \{\delta_a, 1_Q\}$, es decir, $M(A)$ es el grupo con dos elementos $\mathbb{Z}_2$. En conclusión, $S(A) = M(A)$ si y solo si existe una palabra $w \in \Sigma^+$ tal que $\delta_w = 1_Q$.

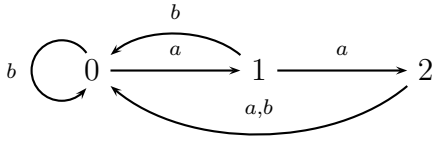
Proposición 1.2.6. *Los semigrupos $S(A)$ y $M(A)$ son finitos.*

Demostración. Q es finito, luego $F_r(Q)$ es finito. ∎

Proposición 1.2.7. *Todo semigrupo finito es el semigrupo asociado a un autómata finito.*

Demostración. Sea S un semigrupo finito. Consideremos en autómata $A(S) = (S^1, S, \delta)$ donde δ viene dada por $\delta(x, y) = xy$. El homomorfismo $a \mapsto \delta_a$ es inyectivo ya que S^1 es un monoide (ver 1.1.13), de donde el resultado sigue. ∎

Ejemplo 1.2.8. Un AF $A = (Q, \Sigma, \delta)$ se llama un **autómata de permutación-reinicio** ("permutation-reset machine") si para todo $a \in \Sigma$, δ_a es biyectiva o una función constante. Consideremos, por ejemplo, el autómata siguiente



δ_a es una permutación y δ_b es una función constante. No es difícil ver que

$$\delta_{a^3} = id, \quad \delta_{a^n b} = \delta_b$$

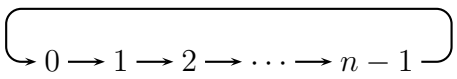
y δ_{ba} y δ_{ba^2} son funciones constantes, de donde deducimos que $S(A)$ es un monoide con 6 elementos: $S(A) = \{\delta_a, \delta_b, \delta_{a^2}, \delta_b, \delta_{ba}, \delta_{ba^2}, \delta_{a^3} = id\}$.

δ	a	b	a^2	ba	ba^2	a^3
0	1	0	2	1	2	0
1	2	0	0	1	2	1
2	0	0	1	1	2	2

Identificando δ_x con x, podemos escribir la tabla de multiplicar de este semigrupo:

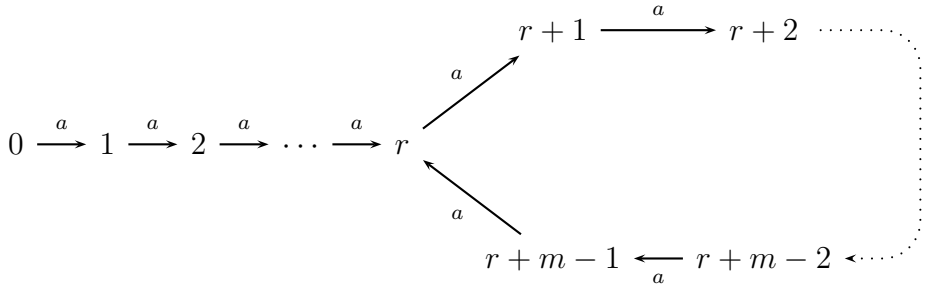
$S(A)$	a	b	a^2	ba	ba^2	a^3
a	a^2	b	a^3	ba	ba^2	a
b	ba	b	ba^2	ba	ba^2	b
a^2	a^3	b	a	ba	ba^2	a^2
ba	ba^2	b	ba^3	ba	ba^2	ba
ba^2	b	b	ba	ba	ba^2	ba^2
a^3	a	b	a^2	ba	ba^2	a^3

Ejemplo 1.2.9. Es fácil ver que el semigrupo del autómata:



es isomorfo a $\mathbb{Z}/n\mathbb{Z}$.

Ejemplo 1.2.10. Es fácil ver que el semigrupo del autómata:



es isomorfo a $C_{m,r}$ (Ejemplo 1.1.9).

Definición 1.2.11. *Un autómata finito $B = (P, \Omega, \varphi)$ decimos que es un* **subautómata** *de $A = (Q, \Sigma, \delta)$ si $P \subseteq Q$, $\Omega \subseteq \Sigma$ y $\varphi \subseteq \delta$ (es decir, $\varphi(p, b) = \delta(p, b)$ para todo $p \in P$, $b \in \Omega$).*

Un caso frecuente es cuando $\Sigma = \Omega$. Entonces un subautómata no es más que un subconjunto de estados cerrado para la acción de Σ^+.

Ejemplo 1.2.12. Sea $A = (Q, \Sigma, \delta)$ un AF. Un estado q se llama *accesible* desde otro estado p si existe $w \in \Sigma^*$ tal que $pw = q$. Fijo un estado q_0 denotemos por Q^a el conjunto de todos los estados accesibles desde q_0. Entonces está claro que (Q^a, Σ, δ') con δ' la restricción de δ a Q^a, es un subautómata de A.

Ejemplo 1.2.13. Sea ρ un relación de equivalencia invariante por la derecha de índice finito sobre Σ^* con Σ finito. Entonces podemos definir el autómata $A(\rho) = (\Sigma^*/\rho, \Sigma, \delta)$ con la acción $[u]_\rho a = [ua]_\rho$. Este autómata tiene la particularidad que dado un estado cualquiera $[u]_\rho$, se tiene $[\epsilon]_\rho u = [u]_\rho$, es decir, todo estado es accesible desde $[\epsilon]_\rho$. Un autómata en el que existe un estado q_0 tal que cualquier otro estado es accesible desde q_0 se llama *cíclico* y el estado q_0 se llama un *generador*.

Proposición 1.2.14. *Sea $B = (P, \Omega, \varphi)$ un subautómata de $A = (Q, \Sigma, \delta)$. Entonces $S(B)$ es imagen homomórfica de un subsemigrupo de $S(A)$.*

Demostración. Sea G el subsemigrupo de $S(A)$ generado por los elementos de la forma δ_b con $b \in \Omega$. Cada elemento de G restringe a una aplicación de P en P. De hecho, δ_w con $w \in \Omega^+$ restringido a P es φ_w. Por tanto tenemos un epimorfismo de G a $S(B)$. ∎

Definición 1.2.15. *Dados dos autómatas $A = (Q, \Sigma, \delta)$ y $B = (P, \Omega, \varphi)$, un homomorfismo de A en B es un par (f, g) con $f : Q \to P$ y $g : \Sigma \to \Omega$ aplicaciones que hacen conmutativo el siguiente diagrama*

$$\begin{array}{ccc} Q & \xrightarrow{f} & P \\ \Big\downarrow{\scriptstyle \delta_a} & & \Big\downarrow{\scriptstyle \varphi_{g(a)}} \\ Q & \xrightarrow{f} & P \end{array}$$

para todo $a \in \Sigma$, es decir, $f[\delta(q, a)] = \varphi[f(q), g(a)]$ para todo $q \in Q$ y $a \in \Sigma$; o bien, si no hay peligro de confusiones, $f(qa) = f(q)g(a)$.

Si $\Sigma = \Omega$ y g es la identidad se dice que tenemos un **homomorfismo de estados** *$f : A \to B$. En este caso la condición de conmutatividad del diagrama puede escribirse simplemente como $f(qa) = f(q)a$ para todo $q \in Q$ y $a \in \Sigma$.*

Si f y g son las dos inyectivas o exhaustivas diremos que tenemos un monomorfismo o un epimorfismo respectivamente.

Definición 1.2.16. *Sea $A = (Q, \Sigma, \delta)$ un autómata. Una relación de equivalencia τ en Q se dice que es una* **relación admisible** *si cumple que $p\ \tau\ q$ implica $\delta(p, a)\ \tau\ \delta(q, a)$ para todo $a \in \Sigma$.*

Una relación admisible τ nos permite definir el **autómata cociente** $A/\tau = (Q/\tau, \Sigma, \delta)$ donde la acción de δ viene dada por $[q]_\tau a = [qa]_\tau$. Se tiene entonces el paso al cociente que es un epimorfismo de estados de A en A/τ.

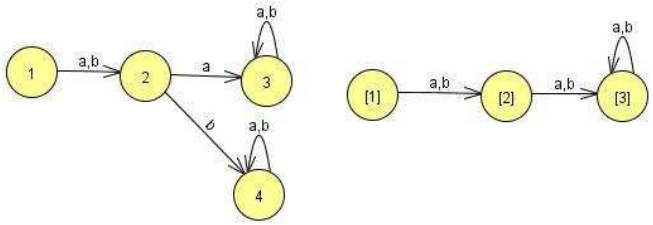


Figura 1.1: Autómata cociente

Ejemplo 1.2.17. Consideremos el autómata de la izquierda en la figura 1.1. La relación dada por la siguiente partición del conjunto de estados es evidentemente admisible

$$[1] = \{1\}, \quad [2] = \{2\}, \quad [3] = \{3, 4\}.$$

El autómata cociente es el que se muestra en la figura de la derecha. No es difícil ver que su semigrupo asociado es $C_{1,2}$ (ver ejemplo 1.1.9), que es imagen homomorfica del semigrupo del autómata inicial (ver la proposición 1.2.20).

Definición 1.2.18. *Sean $A = (Q, \Sigma, \delta)$ y $B = (P, \Omega, \varphi)$ dos autómatas y sea (f, g) un homomorfismo de A en B. Se define la relación* **núcleo** *mediante $q\ \kappa\ q'$ si y solo si $f(q) = f(q')$.*

Proposición 1.2.19. *Con la notación de la definición anterior, la relación núcleo es admisible. Además, si f es sobreyectiva y g biyectiva, entonces el autómata cociente A/κ es isomorfo a B.*

Demostración. Inmediato. ∎

Con respecto al comportamiento de los semigrupos asociados ante homomorfismos de autómatas podemos enunciar el siguiente resultado:

Proposición 1.2.20. *Sean $A = (Q, \Sigma, \delta)$ y $B = (P, \Omega, \varphi)$ dos autómatas. Si B es imagen homomórfica de A, entonces $S(B)$ es imagen homomórfica de $S(A)$.*

DEMOSTRACIÓN. Sea (f, g) un epimorfismo de A en B. La aplicación g se extiende a un epimorfismo entre Σ^+ y Ω^+ que denotaremos de la misma forma. Entonces definimos una aplicación $h : S(A) \to S(B)$ mediante $h(\delta_w) = \varphi_{g(w)}$ con $w \in \Sigma^+$. Veamos que está bien definida. Supongamos que $\delta_u = \delta_v$, entonces $\delta(q, u) = \delta(q, v)$ para todo $q \in Q$, luego como el autómata B es imagen homomórfica de A entonces tenemos que $f(q)g(u) = f(qu) = f(qv) = f(q)g(v)$ para todo $q \in Q$. Como f es exhaustiva, se tiene que $\exists q \in Q$ tal que $f(q) = p$ y se obtiene que $pg(u) = pg(v)$ para todo $p \in P$ y, por tanto, $\varphi_{g(u)} = \varphi_{g(v)}$ como queríamos ver. Finalmente, la aplicación h es epimorfismo al serlo g. ■

Corolario 1.2.21. *Sean A y B dos autómatas. Si B es imagen homomórfica de un subautómata de A, entonces $S(B)$ es imagen homomórfica de un subsemigrupo de $S(A)$.*

DEMOSTRACIÓN. Proposición anterior y proposición 1.2.14. ■

Ejemplo 1.2.22. Consideremos los siguientes autómatas:

Entonces, es fácil ver que el segundo es imagen homomórfica del primero pero no son isomorfos. Sin embargo, los semigrupos asociados son isomorfos.

Capítulo 2

Autómatas y lenguajes formales

En este capítulo se introducen los lenguajes que son aceptados por un autómata finito. Son los llamados lenguajes regulares. Después se demuestra el teorema de Myhill-Nerode para semigrupos y se demuestra que los lenguajes regulares son precisamente los conjuntos reconocibles en el semigrupo finito-generado libre Σ^*. También se considera el autómata minimal que reconoce un determinado lenguaje.

La referencia principal para este capítulo es [2].

2.1. Lenguajes regulares

En este sección definimos lenguaje regular como aquel que es aceptado por un autómata finito y .... Ya que no todos los lenguajes pueden ser reconocidos, como demostramos con el lema de Pumping.

Definición 2.1.1. *Dado un conjunto no vacío Σ, un* ***lenguaje formal*** *con alfabeto Σ es cualquier subconjunto del monoide libre Σ^*. Los elementos de Σ^* se llaman* **palabras** *(en el alfabeto Σ). Si $w = a_1 \cdots a_n$ con $a_i \in \Sigma$, decimos que la* **longitud** *de w es n y lo denotamos por $|w| = n$.*

Definición 2.1.2. *Un* **autómata finito reconocedor***, que abreviamos por AFR, es un quíntupla $A = (Q, \Sigma, q_0, \delta, F)$, donde (Q, Σ, δ) es un autómata finito, $q_0 \in Q$ es un estado llamado* **estado inicial***, y $F \subset Q$ es un conjunto no vacío de estados llamados* **estados finales***.*

Definición 2.1.3. *Sea $A = (Q, \Sigma, q_0, \delta, F)$ un AFR y sea $q \in Q$. Se define el conjunto L_q como el conjunto de palabras de Σ^* que empezando en i terminan en q, es decir,*

$$L_q = \{w \in \Sigma^* \mid q_0 w = q\}.$$

Se define el **lenguaje aceptado por** *A o el* **lenguaje reconocido por** *A como*

$$L(A) = \{w \in \Sigma^* \mid q_0 w \in F\} = \bigcup_{q \in F} L_q.$$

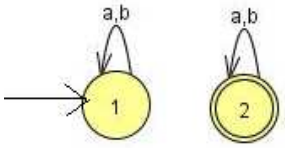


Figura 2.1: Autómata A

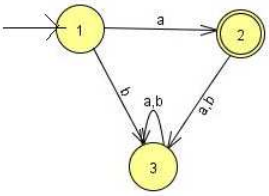


Figura 2.2: Autómata B

Observación. En los grafos que representan a los autómatas se suele representar el estado inicial señalándolo con una flecha o bien nombrándolo como q_0 u otra notación especial. Los estados finales se suelen representar escribiendo en negrita el nombre del estado o rodeando el estado con dos círculos.

Definición 2.1.4. *Un lenguaje formal L con alfabeto finito Σ se dice que es un* ***lenguaje regular*** *si existe un AFR A tal que $L(A) = L$. Denotaremos por $Reg(\Sigma)$ al conjunto de todos los lenguajes regulares con alfabeto Σ.*

Ejemplo 2.1.5. Sea $\Sigma = \{a, b\}$ y consideremos los autómatas dados por las figuras 2.1, 2.2 y 2.3. No es difícil ver que

(a) $L(A) = \emptyset$.

(b) $L(B) = \{a\}$.

(c) $L(C) = \{\text{palabras que terminan en } ab\}$.

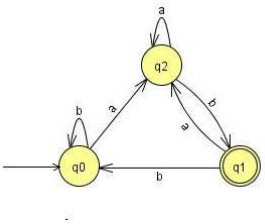


Figura 2.3: Autómata C

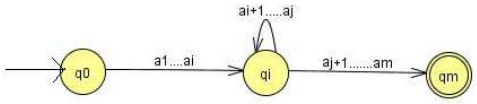


Figura 2.4: Autómata con loop

Proposición 2.1.6. *Existen lenguajes no regulares.*

Demostración. Dado un alfabeto finito Σ, se puede ver que el conjunto de autómatas finitos reconocedores con alfabeto Σ es numerable. Sin embargo, el cardinal de $\mathcal{P}(\Sigma^*)$ es no numerable. ■

El teorema siguiente proporciona una herramienta para identificar lenguajes que no son regulares.

Teorema 2.1.7 (Pumping Lemma)**.** *Sea L un lenguaje regular sobre el alfabeto Σ. Entonces existe un entero $n \geq 1$ tal que para cada $w \in L$ con $|w| \geq n$ podemos encontrar palabras x, y, z de forma que se cumple:*

(a) $w = xyz$.

(b) $|xy| \leq n$.

(c) $|y| \geq 1$.

(d) Para todo $i \geq 0$, se tiene que $xy^iz \in L$

Demostración. El lenguaje L es regular, luego existe un AFR A tal que $L = L(A)$. Sea n el número de estados de A y sea $w = a_1....a_m \in L$ con $m \geq n$. Consideremos la secuencia de estados:

$$q_0, q_1 = q_0a_1, ..., q_m = q_0a_1...a_m.$$

Como $m \geq n$ se sigue que $m + 1 > n$. Pero q_0, q_1, ..., q_n es una lista de estados de longitud $n+1$, así que debe haber alguna repetición en la lista. Sea q_i el primer estado que se repite en la lista y sea $q_j = q_i$ con $0 \leq i < j \leq m$ la primera repetición del estado q_i. Definimos $x = a_1...a_i$, $y = a_{i+1}....a_j$, y $z = a_{j+1}...a_m$. Es claro que las condiciones (a), (b) y (c) se cumplen por construcción y que la (d) se cumple por la existencia de un loop (ver figula 2.4), es decir, un camino que empieza y termina en q_i y que lo puedes repetir el número de veces que quieras antes de llegar a un estado final. ■

Ejemplo 2.1.8. Sea $\Sigma = \{a\}$. Vamos a demostrar que L $=\{a^p \mid p \text{ primo}\}$ no es un lenguaje regular utilizando el teorema anterior.

Supongamos que L es regular. Entonces existe un número n tal que para cada palabra $w \in$L con $|w| \geq n$ se cumplen la 4 consecuencias del lema. Sea p un primo tal que $p > n$. Se tiene que $a^p \in L$ y $|a^p| = p \geq n$ y existen palabras $x, y, z \in \Sigma^*$ tales que (a) $a^p = xyz$, (b) $|xy| \leq n$, (c) $|y| \geq 1$ y (d) $xy^iz \in L$ para todo $i \geq 0$. En particular, $xy^{p+1}z \in L$ y

$$\left|xy^{p+1}z\right| = |x| + \left|y^{p+1}\right| + |z| = |x| + (p+1)\,|y| + |z| = |xyz| + p\,|y|\,.$$

Si $y = a^m$ tenemos

$$\left|xy^{p+1}z\right| = p(m+1.)$$

Pero $p(m+1)$ no es primo ya que $m \geq 1$.

Ejemplo 2.1.9. Sea Sea $\Sigma = \{a, b\}$ y $L = \{a^n b^n : n \geq 1\}$. Entonces L no es regular. Es otra aplicación del teorema 2.1.7 (ver [1, página 46]).

Ejemplo 2.1.10. Es este ejemplo vemos que el recíproco del teorema 2.1.7 no es cierto.

Sea $L \subseteq \{b\}^*$ un lenguaje no regular. Definamos $M = \{a\}^+L \cup \{a\}^*$ (es decir, M está formado por las palabras de la forma a^sb^p con $s \geq 1$ y $b^p \in L$ o bien las palabras pertenecientes $\{a\}^*$). En el ejemplo 3.2.14 se probará que M no es regular. Veamos ahora que, sin embargo, cumple las condiciones del Pumping Lemma con $n = 2$.

Sea $w \in M$ con $|w| \geq n$. Hay 2 posibilidades: $w \in \{a\}^+L$ o bien $w \in \{a\}^*$. Vamos a demostrar la primera, la segunda es fácil. La palabra w es de la forma $w = a^sb^p$ con $s \geq 1$. Elegimos $1 \leq t \leq n, s$ y definimos x, y de forma que $xy = a^t$. Por lo tanto, $z = a^{s-t}b^p$. Está claro que se cumplen la condiciones del Pumping Lemma

2.2. Teorema de Myhill-Nerode

Después de definir los lenguajes regulares como aquellos que son aceptados por un AFR, demostramos en esta sección el teorema de Myhill-Nerode, que los caracteriza como los subconjuntos del monoide Σ^* que son saturados por una congruencia de índice finito.

Definición 2.2.1. *Sea S un semigrupo. Un subconjunto X de S se llama* **reconocible** *si es saturado por una congruencia en S de índice finito. Denotamos por $Rec(S)$ el conjunto formado por todos los subconjuntos de S reconocibles.*

Definición 2.2.2. *Sea S un semigrupo y X un subconjunto de S. Se define la* **congruencia sintáctica** *de X como*

$$s \approx_X t \Leftrightarrow \forall u, v \in S^1(usv \in X \Leftrightarrow utv \in X)$$

Proposición 2.2.3. *Sea S un semigrupo y X un subconjunto de S. La congruencia sintáctica de X es una congruencia en S.*

DEMOSTRACIÓN. Supongamos que $s \approx_X t$ y sea $r \in S$. Entonces, está claro que $usrv \in X \Leftrightarrow utrv \in X$ para todo $u, v \in S^1$. De forma parecida se prueba la invariancia por la izquierda. ■

Definición 2.2.4. *Sea S un semigrupo y X un subconjunto en S. Se define el* ***semigrupo sintáctico de X*** *como el conjunto cociente $S(X) = S/\approx_X$. Si S es un monoide, entonces $S(X)$ también es un monoide y se llama el* ***monoide sintáctico de X****.*

Vamos a definir ahora unas relaciones de equivalencia asociadas a un subconjunto X que se usarán en el teorema de Myhill-Nerode.

Definición 2.2.5. *Sea S un semigrupo y X un subconjunto de S. Se define la* ***equivalencia de Nerode por la derecha****, denotada por N_X^r, de la siguiente forma*

$$sN_X^r t \Leftrightarrow s^{-1}X = t^{-1}X$$

y la ***equivalencia de Nerode por la izquierda****, denotada por N_X^l, como*

$$sN_X^l t \Leftrightarrow Xs^{-1} = Xt^{-1}.$$

donde

$$u^{-1}X = \{v \in S^1 |\ uv \in X\} \quad y \quad Xu^{-1} = \{v \in S^1 |\ vu \in X\}.$$

Proposición 2.2.6. *La relaciones N_X^r y N_X^l son relaciones de equivalencia en S invariantes por la derecha y por la izquierda respectivamente. Además $\approx_X$ es más fina que N_X^r y N_X^l y las tres relaciones $\approx_X, N_X^r, N_X^l$ saturan X.*

DEMOSTRACIÓN. Las relaciones N_X^r, N_X^l son de equivalencia obviamente. La invariancia se deduce de observar que

$$(st)^{-1}X = t^{-1}(s^{-1}X) \quad \text{y} \quad X(st)^{-1} = (Xt^{-1})s^{-1}.$$

Supongamos $s \approx_X t$. Entonces,

$$u \in s^{-1}X \Leftrightarrow su \in X \Leftrightarrow tu \in X \Leftrightarrow u \in t^{-1}X$$

y, por tanto, $s^{-1}X = t^{-1}X$. De forma análoga se demuestra que $\approx_X$ es más fina que N_X^l. Finalmente, ver que la relaciones saturan X no ofrece dificultad. ■

Cuando no haya confusión denotaremos a N_X^r por N_X y la llamaremos la equivalencia de Nerode de X.

Proposición 2.2.7. *La congruencia sintáctica $\approx_X$ es maximal, con respecto a la inclusión, en el conjunto de congruencias que saturan X.*

DEMOSTRACIÓN. Sea θ un congruencia que satura X. Tenemos que probar que si $s\ \theta\ t$, entonces $s \approx_X t$. Si $s\ \theta\ t$ y dados $u, v \in S^1$, por ser θ un congruencia se tiene que $usv\ \theta\ utv$. Además, como θ satura a X se tiene también que

$$usv \in X \Leftrightarrow utv \in X$$

y, en consecuencia, $s \approx_X t$. ■

Proposición 2.2.8. *La equivalencia de Nerode N_X es maximal, con respecto a la inclusión, en el conjunto de las relaciones de equivalencia invariantes por la derecha que saturan X.*

DEMOSTRACIÓN. Sea θ una equivalencia en S invariante por la derecha que satura a X. Supongamos $s\ \theta\ t$. Tenemos $u \in s^{-1}X \Leftrightarrow su \in X$ pero por la invariancia por la derecha, $su\ \theta\ tu$ y, por saturar X, $su \in X \Leftrightarrow tu \in X$. En conclusión $s^{-1}X = t^{-1}X$. ■

Teorema 2.2.9 (Myhill-Nerode)**.** *Sea X un subconjunto de un semigrupo S. Las siguientes condiciones son equivalentes:*

(a) $X \in Rec(S)$.

(b) Existe un semigrupo finito T y un homomorfismo $\phi : S \to T$, tal que $X = \phi^{-1}(\phi(X))$.

(c) Un semigrupo sintactico $S(X)$ es finito.

(d) X es un conjunto saturado por una equivalencia invariante por la derecha de indice finito.

(e) La equivalencia de Nerode N_X tiene indice finito.

DEMOSTRACIÓN. $(a) \Rightarrow (b)$ Sea θ una congruencia de índice finito que satura X y sea $\phi : S \to S/\theta$ el epimorfismo canónico. Dado que para cualquier $s \in S$ se tiene $[s]_\theta = \phi^{-1}(\phi(s))$ y que X es unión de clases de equivalencia, deducimos $X = \phi^{-1}(\phi(X))$.

$(b) \Rightarrow (c)$ Sea T un semigrupo finito y $\phi : S \to T$ un homomorfismo tal que $X = \phi^{-1}(\phi(X))$. Sea κ la congruencia núcleo de ϕ. Entonces, por el teorema 1.1.19, tenemos un monomorfismo

$$\hat{\phi} : S/\kappa \to T.$$

Dado que T es finito, deducimos que la congruencia κ es de índice finito. Observemos también que

$$y \in [x]_\kappa \Rightarrow \phi(y) = \phi(x) \Rightarrow y \in \phi^{-1}(\phi(x)) \subseteq X$$

y, por tanto, κ satura X. Por la proposición 2.2.7, la congruencia κ es más fina que la congruencia sintáctica, por lo que esta debe ser también de índice finito (proposición 1.1.21).

$(c) \Rightarrow (d)$ $S(X)$ es finito y, por tanto, $\approx_X$ es de índice finito. Dado que $\approx_X$ es invariante por la derecha y satura X tenemos (d).

$(d) \Rightarrow (e)$ Sea θ una equivalencia invariante por la derecha de índice finito que satura X. Por la proposición 2.2.8 debe ser $\theta \subseteq N_X$, de donde, otra vez por la proposición 1.1.21, tenemos que N_X es de índice finito.

$(e) \Rightarrow (a)$ Vamos a demostrar que la congruencia sintáctica tiene índice finito. Sea θ la relación

$$t_1 \ \theta \ t_2 \Leftrightarrow \forall s \in S^1, \ [st_1]_{N_X} = [st_2]_{N_X}.$$

Ahora observemos que

$$t_1 \ \theta \ t_2 \Leftrightarrow \forall s \in S^1, st_1 \ N_X \ st_2 \Leftrightarrow \forall s \in S^1, (st_1)^{-1}X = (st_2)^{-1}X \Leftrightarrow$$

$$\forall s, u \in S^1, (st_1u \in X \Leftrightarrow st_2u \in X) \Leftrightarrow t_1 \approx_X t_2.$$

Es decir, la congruencia θ coincide con la congruencia sintáctica. Por hipótesis, N_X tiene índice finito. Sea $\{s_1, \ldots, s_r\}$ un conjunto de representantes de las clases de equivalencia módulo N_X y consideremos la siguiente aplicación

$$S/\theta \longrightarrow S/N_X \times \cdots \times S/N_X$$

$$[t]_\theta \mapsto ([t]_{N_X}, [s_1t]_{N_X}, \ldots, [s_rt]_{N_X}).$$

La aplicación está bien definida y vamos a probar que es inyectiva, con lo cual θ y, por tanto, $\approx_X$ tendrá índice finito. Supongamos que $[t]_\theta$ y $[t']_\theta$ tienen la misma imagen. Entonces,

$$t \ N_X \ t', \quad s_1t \ N_X \ s_1t', \ \ldots, \ s_rt \ N_X \ s_rt'.$$

Dado $s \in S$, existe un i tal que $s \ N_X \ s_i$ y entonces

$$st \ N_X \ s_it, \quad s_it \ N_X \ s_it', \quad s_it' \ N_X \ st',$$

es decir, $[st]_{N_X} = [st']_{N_X}$ para todo $s \in S^1$ y, por tanto, $[t]_\theta = [t']_\theta$. ∎

Observación. Con una demostración análoga, podemos ver que todas las condiciones del teorema anterior son equivalentes también a que la equivalencia de Nerode por la izquierda tenga índice finito

Corolario 2.2.10. *Rec(S) es cerrado bajo las operaciones booleanas.*

Demostración. Sea $X \in \mathrm{Rec}(S)$. La congruencia sintáctica de X coincide con la congruencia sintáctica de X^c. Por tanto, $S(X) = S(X^c)$.

Supongamos ahora que $X_1, X_2 \in \mathrm{Rec}(S)$. Sean θ_1 y θ_2 dos congruencias de índice finito que saturan X_1 y X_2 respectivamente. Sea $\theta = \theta_1 \cap \theta_2$ y m_1, m_2 los índices de

θ_1 y θ_2 respectivamente. Entonces, por la proposición 1.1.17, el índice de θ es menor o igual que $m_1 m_2$. Como es fácil ver que θ satura a X_1 y X_2, entonces también satura a su intersección $X_1 \cap X_2$. Por lo tanto $X_1 \cap X_2 \in \mathrm{Rec}(S)$.

Finalmente, dado que $X_1 \cup X_2 = (X_1^c \cap X_2^c)^c$, obtenemos $X_1 \cup X_2 \in \mathrm{Rec}(S)$. ∎

En el siguiente teorema demostramos uno de los objetivos fundamentales de este trabajo: los lenguajes regulares en el alfabeto Σ son los conjuntos reconocibles del semigrupo Σ^*.

Teorema 2.2.11. *Sea Σ un alfabeto finito. Entonces, $L \in Reg(\Sigma)$ si y solo si $L \in Rec(\Sigma^*)$.*

Demostración. Sea $L \subseteq \Sigma^*$ un lenguaje regular y sea $A = (Q, \Sigma, \delta, q_0, F)$ un AFR tal que $L(A) = L$. Definimos en Σ^* la relación de equivalencia definida como: para todo $u, v \in \Sigma^*$

$$u \theta v \Leftrightarrow q_0 u = q_0 v.$$

La relación θ es claramente invariante por la derecha, y si $u \in L$ y $u\theta v$, se tiene $q_0 u = q_0 v \in F$ de donde $v \in L$ y θ satura a L. La aplicación

$$\Sigma^*/\theta \to Q$$

definida por $[u]_\theta \mapsto iu$ es claramente inyectiva, luego θ es también de índice finito. En consecuencia, por el teorema de Myhill-Nerode, tenemos que $L \in \mathrm{Rec}(\Sigma^*)$.

Recíprocamente, supongamos que $L \in \mathrm{Rec}\,(\Sigma^*)$. Por el teorema de Myhill-Nerode el índice de la equivalencia de Nerode N_L es finita. Definimos el autómata

$$A_L = (\Sigma^*/N_L, \Sigma, [\epsilon]_{N_L}, \delta, F),$$

donde la función de transición δ está definida como: para todo $u \in \Sigma^*$ y $a \in \Sigma$

$$[u]_{N_L} a = [ua]_{N_L}$$

y F=$\{[u]_{N_L} | u \in L\}$. La función δ está bien definida porque N_L es invariante por la derecha. Veamos que $L = L(A)$. Sea $u \in L(A_L)$, entonces $[\epsilon]_{N_L} u = [u]_{N_L} \in F$, es decir, $[u]_{N_L} = [v]_{N_L}$ con $v \in L$, pero como N_L satura L se tiene que $u \in L$. Recíprocamente, si $u \in L$, entonces $[\epsilon]_{N_L} u = [u]_{N_L} \in F$. ∎

Definición 2.2.12. *El autómata A_L de la demostración del teorema anterior se llama* **autómata de Nerode**.

Veremos en la sección siguiente que el autómata de Nerode es el autómata minimal que reconoce a L. Veamos a continuación que el semigrupo de A_L coincide con el semigrupo sintáctico de L.

Proposición 2.2.13. *Dado $L \in Reg(\Sigma)$, el semigrupo sintáctico $S(L)$ coincide con el semigrupo del autómata de nerode $S(A_L)$.*

Demostración. Dado que $S(A_L)$ es isomorfo a $\Sigma^*/\sim$ con $\sim$ la relación $u \sim v \Leftrightarrow \delta_u = \delta_v$ (ver comentario después de la definición 1.2.4), bastará ver que la relación $\sim$ coincide con la congruencia sintáctica. En efecto:

$$u \sim v \Leftrightarrow [x]_{N_L} u = [x]_{N_L} v,\ \forall x \in \Sigma^* \Leftrightarrow xu\ N_L\ xv\ \forall x \in \Sigma^* \Leftrightarrow u \approx_X u$$

según se vio en la demostración de (e)⇒(a) del teorema 2.2.9. ∎

Corolario 2.2.14. *Los lenguajes finitos son regulares.*

Demostración. Dado un lenguaje finito, no es difícil construir un autómata que lo reconozca. Pero dado que Reg (Σ) es cerrado bajo operaciones booleanas por el corolario 2.2.10 y el teorema anterior, es suficiente probar que el lenguaje $L = \{w\}$ con $w \in \Sigma^*$ es regular, lo cual ya se hizo en el ejemplo 2.1.5. ∎

Ejemplo 2.2.15. Sea $\Sigma = \{0, 1\}$. Veamos que el lenguaje $L = \{0^n 10^n \mid n \in \mathbb{N}\} \subseteq \Sigma^*$ no es regular. Sean p, q dos naturales distintos. Si $0^p \approx_L 0^q$, entonces $x0^p y \in L \Leftrightarrow x0^q y \in L$. En particular, si tomamos $x = \epsilon$ e $y = 10^q$ tendríamos $0^p 10^q \in L \Leftrightarrow 0^q 10^q \in L$ lo cual es contradictorio. En conclusión, las palabras $0, 0^2, 0^3, \ldots$ representan clases distintas por la congruencia sintáctica, lo cual contradice el teorema 2.2.9.

2.3. Autómata minimal

Definición 2.3.1. *Sea L un lenguaje regular en el alfabeto Σ. Un AFR A con alfabeto Σ se dice* **minimal para** *L, si $L(A) = L$ y si B es otro AFR con alfabeto Σ que reconoce a L, entonces el número de estados de A es menor o igual que el número de estados de B.*

En esta sección se demuestra que el autómata de Nerode es minimal y que todos los autómatas minimales que reconocen un lenguaje L son isomorfos. Asimismo se proporciona una descripción alternativa del autómata de Nerode, en concreto, como el autómata reducido y accesible que reconoce a L, lo cual ofrece un método práctico de cálculo.

Observación. Dado un autómata A, denotaremos en esta sección por $|A|$ al número de estados de A.

Definición 2.3.2. *Sea $A = (Q, \Sigma, q_0, \delta, F)$ un autómata finito. Decimos que un estado $q \in Q$ es* **accesible** *si hay una palabra $x \in \Sigma^*$ tal que $q_0 \cdot x = q$. Un estado no accesible se dice* **inaccesible**. *Un autómata se dice que es* **accesible** *si cada estado es accesible.*

Como ya vimos en el ejemplo 1.2.12 los estados inaccesibles pueden ser eliminados sin mayores problemas:

Proposición 2.3.3. *Dado un AFR A, existe un autómata accesible A^a tal que $L(A) = L(A^a)$.*

DEMOSTRACIÓN. Sea $A = (Q, \Sigma, q_0, \delta, F)$ un autómata finito. Definimos A^a como el subautómata

$$A^a = (Q^a, \Sigma, q_0, \delta^a, F^a)$$

con $Q^a = \{q \in Q \mid q \text{ accesible}\}$, $F^a = F \cap Q^a$, δ^a es la restricción de δ a Q^a. Es evidente que $L(A) = L(A^a)$. ∎

Proposición 2.3.4. $|A^a| \leq |A|$ y $|A^a| = |A|$ si y solo si A es accesible.

DEMOSTRACIÓN. Evidente. ∎

Definición 2.3.5. *Dado un autómata $A = (Q, \Sigma, q_0, \delta, F)$, definimos la siguiente relación de equivalencia*

$$p \simeq_A q \Leftrightarrow \forall x \in \Sigma^* \ (px \in F \Leftrightarrow qx \in F).$$

En este caso, diremos que p, q son **indistinguibles**. *Si se cumple que $p \simeq_A q \Leftrightarrow p = q$ diremos que A es* **reducido**.

Proposición 2.3.6. *Dado un AFR A, existe un autómata reducido A^r tal que $L(A) = L(A^r)$.*

DEMOSTRACIÓN. Sea $A = (Q, \Sigma, q_0, \delta, F)$ un autómata finito. La relación $\simeq_A$ es claramente admisible (ver definición 1.2.16) y podemos considerar el autómata cociente. Definimos pues A^r como este cociente con estado inicial $[q_0]_\simeq$, estados finales $[F]_\simeq = \{[q]_\simeq \mid q \in F\}$ y la función de transición obvia. Veamos en primer lugar que A^r es reducido. Si denotamos a $[q]_\simeq$ simplemente por $[q]$, observemos en primer lugar que $[q] \in [F]$ si y solo si $q \in F$, pues si $[q] = [p]$ con $p \in F$, entonces $px \in F \Leftrightarrow qx \in F$ para todo x, en particular para $x = \epsilon$. Supongamos $[p] \simeq [q]$. Tenemos

$$px \in F \Leftrightarrow [p]x = [px] \in [F] \Leftrightarrow [qx] \in [F] \Leftrightarrow qx \in F$$

y, por tanto, $[p] = [q]$.

Finalmente, ver que A y A^r reconocen el mismo lenguaje no ofrece ninguna dificultad. ∎

Proposición 2.3.7. $|A^r| \leq |A|$ y $|A^r| = |A|$ si y solo si la relación $\simeq_A$ es la identidad, es decir, A es reducido.

DEMOSTRACIÓN. Evidente. ■

Proposición 2.3.8. *Sea A un autómata finito reconocedor. Entonces hay un autómata accesible y reducido A^{ar} tal que $L(A) = L(A^{ar})$.*

DEMOSTRACIÓN. Se deduce de las proposiciones 2.3.3 y 2.3.6 que el autómata buscado es $(A^a)^r$. ■

En la definición 1.2.15 se definió el concepto de homomorfismo de autómatas. Necesitamos ahora ampliar esta definición a los AFR, aunque nos restringiremos a un isomorfismo de estados.

Definición 2.3.9. *Sean $A = (Q, \Sigma, q_0, \delta, F)$ y $B = (P, \Sigma, p_0, \gamma, T)$ dos AFR. Un* **isomorfismo** *$g : A \to B$ es una aplicación $g : Q \to P$ biyectiva tal que*

(a) $g(\delta(q, a)) = \gamma(g(q), a), \ \ \forall q \in Q, a \in \Sigma.$

(b) $g(q_0) = p_0.$

(c) $g(F) = T.$

Proposición 2.3.10. *Sean $A = (Q, \Sigma, q_0, \delta, F)$ y $B = (P, \Sigma, p_0, \gamma, T)$ dos AFR isomorfos. Entonces $L(A) = L(B)$.*

DEMOSTRACIÓN. Sea $g : A \to B$ un isomorfismo y denotemos por un punto la acción de δ y γ. Entonces,

$$u \in L(A) \Leftrightarrow iu \in F \Leftrightarrow g(q_0 u) = p_0 u \in T \Leftrightarrow u \in L(B)$$

como queríamos ver. ■

En los resultados que siguen vamos a ver que todo autómata minimal para L es accesible y reducido y que todos los autómatas accesibles y reducidos que lo reconocen son minimales para L y son isomorfos al autómata de Nerode de ese lenguaje.

Proposición 2.3.11. *Sea L un lenguaje regular. Si A es un AFR minimal para L, entonces A es accesible y reducido.*

DEMOSTRACIÓN. Evidente a partir de las proposiciones 2.3.4 y 2.3.7. ■

Proposición 2.3.12. *Sea L un lenguaje regular. El autómata de Nerode A_L es minimal para L.*

DEMOSTRACIÓN. Sea $A = (Q, \Sigma, q_0, \delta, F)$ un AFR tal que $L(A) = L$. En la demostración del teorema 2.2.11 se considera la relación de equivalencia en Σ^* dada por $u\theta v \Leftrightarrow q_0 u = q_0 v$ y se observa que es invariante por la derecha y satura a L. También se considera la aplicación

$$\Sigma^*/\theta \to Q$$

definida por $[u]_\theta \mapsto q_0 u$ que es inyectiva. En consecuencia, el índice de θ es finito y, por tanto, por las proposiciones 1.1.21 y 2.2.8, el índice de la equivalencia de Nerode es menor o igual que el cardinal de Q. En definitiva $|A_L| \leq |A|$. ■

Teorema 2.3.13. *Sea L un lenguaje regular y sea A un AFR accesible y reducido tal que $L(A) = L$. Entonces, A es isoformo al autómata de Nerode de L y, por tanto, es minimal para L.*

DEMOSTRACIÓN. Sea $A = (Q, \Sigma, q_0, \delta, F)$ accesible y reducido con $L(A) = L$. Definimos la aplicación

$$g : \Sigma^*/N_L \to Q$$

por $g([u]_{N_L}) = q_0 u$. Veamos en primer lugar que está bien definida. Supongamos que $u\ N_L\ v$. Entonces, para todo $x \in \Sigma^*$ tenemos

$$q_0 ux \in F \Leftrightarrow ux \in L \Leftrightarrow vx \in L \Leftrightarrow q_0 vx \in F.$$

En consecuencia, $q_0 u$ es indistinguible de $q_0 v$ y como A es reducido, debe ser $q_0 u = q_0 v$.

Dado que A es accesible, la aplicación g es suprayectiva y como el cardinal de Σ^*/N_L es menor o igual que el cardinal de Q por la minimalidad del autómata de Nerode, tenemos que g es biyectiva. El resto de propiedades que hacen de g un isomorfismo de estados son prácticamente evidentes. ■

Ejemplo 2.3.14. Consideremos el autómata A de la figura 2.5. Vamos a hallar el autómata minimal que reconoce el mismo lenguaje. Para ello transformaremos este autómata en otro accesible y reducido.

Para empezar eliminamos los estados no accesibles. Es fácil ver que el estado número 8 no es accesible pero sí los son los demás.

Ahora hallaremos las clases de equivalencia módulo la relación de la definición 2.3.5. Para empezar separamos los estados finales de los no finales: $F = \{3, 4, 7\}$ y $F^c = \{1, 2, 5, 6\}$. Está claro que un estado de F nunca puede estar relacionado con un estado de F^c. Observamos fácilmente a partir del grafo del autómata que todos los estados finales son indistinguibles entre sí, pues si p es final, pu es final para todo $u \in \Sigma^*$.

Hagamos ahora una tabla que muestre para cada pareja de estados (p, q) su imagen por a o b, es decir (px, qx) con $x = a, b$ y, naturalmente, solo para los estados de F^c.

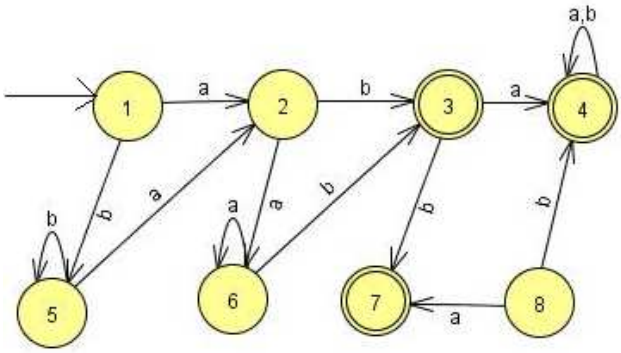


Figura 2.5: Autómata no minimal

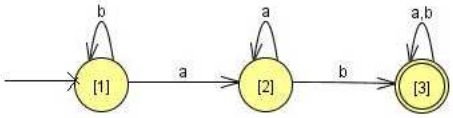


Figura 2.6: Autómata minimal

δ	$(1,2)$	$(1,5)$	$(1,6)$	$(2,5)$	$(2,6)$	$(5,6)$
a	$(2,6)$	$(2,2)$	$(2,6)$	$(6,2)$	$(6,6)$	$(2,6)$
b	$(5,3)$	$(5,5)$	$(5,3)$	$(3,5)$	$(3,3)$	$(5,3)$

Podemos observar en la tabla que los estados 1 y 5 tienen la misma imagen para las 2 letras. Por lo tanto, los estados 1 y 5 son indistinguibles pues $1x = 5x$ para todo $x \in \Sigma^*$. Lo mismo pasa para los estados 2 y 6. Los estados 1 y 2 son distinguibles porque $1b = 2$ y $2b = 3$, y $2 \in F^c$ y $3 \in F$.

En conclusión, tenemos que las clases de equivalencia de $\simeq_A$ son $[1] = \{1, 5\}$, $[2] = \{2, 6\}$, $[3] = \{3, 4, 7\}$, y el autómata A^{ar} es el correspondiente a la figura 2.6.

Capítulo 3

Expresiones racionales

En el capítulo anterior se ha probado que los lenguajes regulares son los conjuntos reconocibles por el semigrupo Σ^*. Se ha definido el semigrupo sintáctico de un lenguaje y se ha probado que coincide con el semigrupo del autómata minimal que lo reconoce, es decir, el autómata de Nerode.

En este último capítulo introducimos los autómatas no deterministas y los usamos para probar que los lenguajes dados por expresiones racionales también son regulares. Finalmente, el teorema de Kleene, nos permite indentificar los lenguajes regulares con los dados por expresiones racionales y proporciona un método de cálculo de la expresión racional basado en la resolución de un particular sistema de ecuaciones.

Las referencias para este capítulo son [1] y [4].

3.1. Autómatas no deterministas

Hemos estado trabajando hasta ahora con autómatas finitos deterministas, es decir, dado el estado q y una letra a, el autómata evoluciona hacia otro estado unívocamente determinado. En esta sección introducimos los autómatas no deterministas. En ellos $\delta(q, a)$ puede ser un conjunto de estados incluido el conjunto vacío. Introduciremos también los autómatas con $\epsilon-$transiciones, en los que podemos además pasar de un estado a otro con "coste cero", es decir, sin que sea necesario ningún "input". Demostraremos que los lenguajes que aceptan siguen siendo regulares.

Definición 3.1.1. *Un AFR* **no determinista** *es un quíntupla $A = (Q, \Sigma, q_0, \delta, F)$, donde Q es un conjunto finitos de estados, Σ es un alfabeto finito, $q_0 \in Q$ es el estado inicial, $F \subseteq Q$ es el conjunto de estados finales y δ es una aplicación*

$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$

donde $\mathcal{P}(Q)$ denota el conjunto de las partes de Q

Si denotamos por un punto la acción de δ, tenemos que en un AFR no determinista $qa \subseteq Q$, es decir, qa no es un estado sino un conjunto de estados que incluso puede ser vacío.

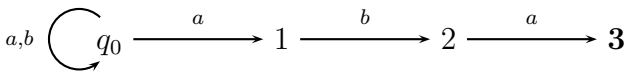
La aplicación δ se extiende a una función definida en $Q \times \Sigma^*$, que denotaremos de la misma manera, mediante

$$\begin{cases} \delta(q, \epsilon) = q, \ q \in Q \\ \\ \delta(q, ab) = \bigcup_{p \in \delta(q,a)} \delta(p, b), \ q \in Q, \ a, b \in \Sigma \end{cases}$$

Definición 3.1.2. *Se define el lenguaje aceptado por un AFR no determinista A como*

$$L(A) = \{u \in \Sigma^* \mid q_0 u \cap F \neq \emptyset\}.$$

Ejemplo 3.1.3. El autómata cuyo grafo es



es un autómata no determinista. Tenemos, por ejemplo, $q_0 a = \{q_0, 1\}$ y $1a = \emptyset$. No es difícil ver que el lenguaje que acepta son las palabras que terminan en aba.

Veamos ahora que el lenguaje aceptado por un AFR no determinista es regular.

Proposición 3.1.4. *Sea L un lenguaje aceptado por un autómata finito no determinista. Entonces L es regular.*

Demostración. Sea $A = (Q, \Sigma, \delta, q_0, F)$ sea un AFR no determinista que acepta L. Definimos un AFR $A' = (Q', \Sigma, \delta', q_0', F')$, como sigue $Q' = \mathcal{P}(Q)$, el conjunto de estados finales F' está formado por los conjuntos de estados que contiene un estado final de A, $q_0' = \{q_0\}$ y definimos

$$\delta'(\{q_1, ..., q_r\}, a) = \bigcup_{j=1,...,r} \delta(q_j, a).$$

Si $a, b \in \Sigma$, se tiene

$$\delta'(\{q_0\}, ab) = \delta'(\delta'(\{q_0\}, a), b) = \delta'(\delta(q_0, a), b) = \bigcup_{p \in \delta(q_0,a)} \delta(p, b) = \delta(q_0, ab)$$

lo que permite ver fácilmente que $L(A) = L(A')$ ([1, página 63] o [4, página 22]). ∎

A continuación vamos a introducir otra clase de autómatas no deterministas

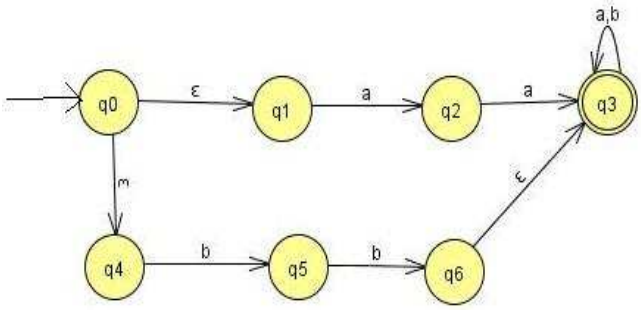


Figura 3.1: ϵ-transiciones

Definición 3.1.5. *Un* ***AFR no determinista con*** ϵ***-transiciones*** *es una quíntupla* $A = (Q, \Sigma, \delta, q_0, T)$ *donde*

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$$

y los demás elementos están definidos como en la 3.1.1.

Ejemplo 3.1.6. Consideremos el autómata con ϵ- transiciones dado por la figura 3.1. Tenemos $q_0\epsilon = \{q_1, q_4\}$. La idea es que estar en q_0 equivale a estar simultáneamente en q_1 y q_4. Con esta idea, las palabras aa y bb son del lenguaje aceptado. De hecho, el lenguaje aceptado es precisamente el conjunto de todas las palabras que empiezan por aa o bb.

Definición 3.1.7. *Dado un AFR no determinista con* ϵ*-transiciones* $A = (Q, \Sigma, \delta, q_0, T)$ *y dado* $p \in Q$ *se define la* ***ϵ-clausura*** *de* p *como el conjunto de estados* $E(p)$ *dado por la condiciones*

(a) $p \in E(p)$

(b) Si $q \in E(p)$*, entonces* $\delta(q, \epsilon) \subseteq E(p)$*.*

Definición 3.1.8. *Dado un AFR no determinista con* ϵ*-transiciones* $A = (Q, \Sigma, \delta, q_0, F)$*, definimos la función* $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$ *de la forma siguiente:*

(a) $\hat{\delta}(p, \epsilon) = E(p)$*.*

(b) $\hat{\delta}(p, a) = E(E(p)a), \;\; a \in \Sigma$*.*

(c) $\hat{\delta}(p, ab) = \bigcup_{q \in \hat{\delta}(p,a)} \hat{\delta}(q, b), \;\; a, b \in \Sigma$*.*

Ejemplo 3.1.9. Para el ejemplo de la figura 3.1 tenemos $E(q_0) = \{q_0, q_1, q_4\}$, $E(q_1) = \{q_1\}$, $E(q_6) = \{q_6, q_3\}$, $\hat{\delta}(q_0, a) = \{q_2\}$, $\hat{\delta}(q_6, ab) = \{q_3\}$.

Definición 3.1.10. *Dado un AFR no determinista con ϵ-transiciones $A = (Q, \Sigma, \delta, q_0, F)$, se define el* **lenguaje aceptado por** *A como*

$$L(A) = \left\{ u \in \Sigma^* \mid \hat{\delta}(q_0, u) \cap F \neq \emptyset \right\}.$$

Proposición 3.1.11. *Si L es aceptado por un autómata con ϵ-transiciones, entonces L es regular.*

Demostración. Sea $A = (Q, \Sigma, \delta, q_0, F)$ un autómata no determinista con ϵ-transiciones. Vamos a construir un autómata no determinista sin ϵ-transiciones que reconozca el mismo lenguaje.

$A' = (Q, \Sigma, \delta', q_0, F')$ el AFR no determinista dado por

$$\delta'(q, a) = \hat{\delta}(q, a),\ q \in Q,\ a \in \Sigma$$

$$F' = \begin{cases} F \cup \{q_0\} & \text{si } E(q_0) \cap F \neq \emptyset \\ F & \text{en otro caso} \end{cases}$$

Veamos que $\delta'(q_0, x) = \hat{\delta}(q_0, x)$ para toda $x \in \Sigma^+$ (para $x = \epsilon$ evidentemente no coinciden. Procediendo por inducción sobre la longitud de x, si $|x| = 1$, el resultado es cierto por la propia definición. Supongamos $|x| > 1$ y pongamos $x = wa$ con $a \in \Sigma$. Entonces tenemos

$$\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a) = \delta'(\hat{\delta}(q_0, w), a) = \bigcup_{q \in \hat{\delta}(q_0, w)} \delta'(q, a) = \bigcup_{q \in \hat{\delta}(q_0, w)} \hat{\delta}(q, a) = \hat{\delta}(q_0, wa).$$

Para completar la demostración, debemos ver que $\delta'(q_0, x)$ contiene un estado de F' si y solo si $\hat{\delta}(q_0, x)$ contiene un estado de F. Para $x = \epsilon$, la afirmación es consecuencia inmediata de la definición de δ y $\hat{\delta}$. Si $x \neq \epsilon$, entonces $x = wa$ con $a \in \Sigma$. Si $\hat{\delta}(q_0, x)$ contiene un estado de F, entonces está claro que $\delta'(q_0, x)$ contiene el mismo estado en F'. Recíprocamente, si $\delta'(q_0, x)$ contiene un estado en F' distinto de q_0, entonces $\hat{\delta}(q_0, x)$ contiene un estado en F. Si si $\delta'(q_0, x)$ contiene a q_0, y q_0 no está en F, entonces como $\hat{\delta}(q_0, x) = E(\delta(\hat{\delta}(q_0, w), a))$, el estado en $E(q_0)$ y en F debe estar en $\hat{\delta}(q_0, x)$. ∎

Ejemplo 3.1.12. Apliquemos el procedimiento anterior al autómata de la figura 3.1. Tenemos un autómata no determinista con ϵ-transiciones y queremos convertirlo en un autómata sin ϵ-transiciones que reconozca el mismo lenguaje. Llamemos A' al autómata buscado. Podemos hacer la siguiente tabla que se explica por sí misma y obtenemos el autómata de la figura 3.2.

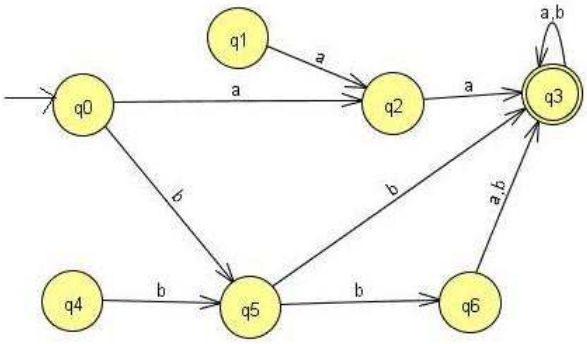


Figura 3.2: Sin ϵ-transiciones

A'	$E(*)$	$E(*)\cdot a$	$E(*)\cdot b$	$E(E(*)\cdot a)$	$E(E(*)\cdot b)$
q_0	$\{q_0, q_1, q_4\}$	$\{q_2\}$	$\{q_5\}$	$\{q_2\}$	$\{q_5\}$
q_1	$\{q_1\}$	$\{q_2\}$	$\emptyset$	$\{q_2\}$	$\emptyset$
q_2	$\{q_2\}$	$\{q_3\}$	$\emptyset$	$\{q_3\}$	$\emptyset$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$
q_4	$\{q_4\}$	$\emptyset$	$\{q_5\}$	$\emptyset$	$\{q_5\}$
q_5	$\{q_5\}$	$\emptyset$	$\{q_6\}$	$\emptyset$	$\{q_3, q_6\}$
q_6	$\{q_3, q_6\}$	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$

3.2. El teorema de Kleene

En esta sección veremos que para un semigrupo finito generado libre S, la familia de los subconjuntos reconocibles $\mathrm{Rec}\,(S)$ y la familia de los subconjuntos racionales $\mathrm{Rat}\,(S)$ que se definirán continuación, coinciden. Es el teorema de Kleene. Para demostrar este teorema utilizaremos los autómatas no deterministas por un lado y las ecuaciones lineales de lenguajes por otro.

Definición 3.2.1. *Dado un semigrupo S, se llaman* **operaciones racionales** *en $\mathcal{P}(S)$ a las siguientes:*

- *La unión, que denotará por $+$.*
- *El producto, definido como $A\cdot B = \{a\cdot b\,|a\in A, b\in B\}$, $A, B\in P(S)$ (ver ejemplo 1.1.11).*
- *Dado $A\in P(S), A^+ = A + A^2 + \cdots + A^n + \cdots$, es decir, A^+ es el subsemigrupo generdo por A.*

Si S es un monoide y $A\in\mathcal{P}(S)$, el submonoide generado por A se denota por A^* y se obtiene con las operaciones anteriores como $A^* = A^+ + 1$.

Proposición 3.2.2. *Si S es un monoide y $A, B, C \in \mathcal{P}(S)$, se tiene*

- $A(B+C) = AB + AC, \quad (B+C)A = BA + CA.$
- $AA^+ = A^+A.$
- $A^+ = A^*A.$
- $A^+ = (A + \cdots + A^p)(A^p)^*, p > 1.$
- $(A^*B)^+ = (A+B)^*B.$
- $(A+B)^+ = (A^*B)^+A^* + A^+.$
- $(AB)^+ = A(BA)^*B.$

DEMOSTRACIÓN. Fácil. ■

Definición 3.2.3. *Dado un semigrupo S se define el conjunto $Rat(S)$ como más pequeño de los subconjuntos de $\mathcal{P}(S)$ que contiene a $\emptyset$ y a los conjuntos finitos y que es cerrado para las operaciones racionales antes definidas .*

Proposición 3.2.4. *$Rat(S)$ coincide con el conjunto formado por los elementos $A \in \mathcal{P}(S)$ construidos de forma inductiva de la siguiente manera: $A = \emptyset$ o $A = \{s\}$ con $s \in S$, o bien A se obtiene de los anteriores en un número finito de pasos por la aplicación de operaciones racionales.*

DEMOSTRACIÓN. Sea A un subconjunto de S construido como se indica en el enunciado. Entonces, A pertenece a cualquier subconjunto de $\mathcal{P}(S)$ que contenga al $\emptyset$ y a los conjuntos finitos y sea cerrado para las operaciones racionales. Esto prueba una inclusión. La otra inclusión es parecida. ■

En el caso $S = \Sigma^*$ con Σ un alfabeto finito, los elementos de $\mathrm{Rat}\,(\Sigma^*)$ se llaman **expresiones racionales en alfabeto** Σ y tradicionalmente se definen recursivamente por las reglas siguientes:

(a)) $\emptyset$ es una expresión racional que denota el conjunto vacío.

(b) ϵ es una expresión racional que denota el conjunto $\{\epsilon\}$.

(c) Para cada $s \in \Sigma$, s es una expresión racional que denota el conjunto $\{s\}$.

(d) Si α y β son expresiones racionales que denotan los conjuntos A y B, entonces $\alpha + \beta$, $\alpha\beta$, α^+ y α^* son expresiones racionales que denotan los conjuntos $A \cup B$, $A \cdot B$, A^+ y A^* respectivamente.

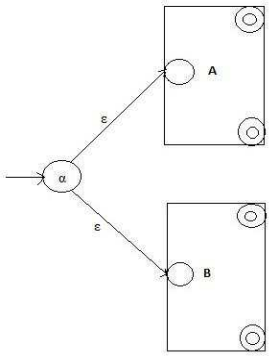


Figura 3.3: Suma de lenguajes

Ejemplo 3.2.5. Si $\Sigma = \{0, 1\}$, las siguientes expresiones son expresiones racionales en el alfabeto Σ:

- $(0 + 1)^*010$ son las palabras terminadas en 010.
- $(0+1)^*000(0+1)^*$ son las palabras que contienen al menos tres ceros consecutivos.

Vamos ahora a demostrar $Rat\,(\Sigma^*) \subseteq \mathrm{Reg}\,(\Sigma)$, que forma parte del enunciado del teorema de Kleene. Para ello utilizaremos autómatas no deterministas con ϵ-trasiciones.

Proposición 3.2.6. *Sea Σ un conjunto finito. Entonces $Rat\,(\Sigma^*) \subseteq Reg(\Sigma)$.*

Demostración. Sabemos que el $\emptyset$ y los conjuntos finitos son regulares (corolario 2.2.14). Hay que demostrar que $\mathrm{Reg}(\Sigma)$ es cerrado para operaciones racionales, es decir, si $L_1, L_2 \in \mathrm{Reg}\,(\Sigma)$, entonces tenemos que demostrar que $L_1 + L_2, L_1L_2, L_1^* \in \mathrm{Reg}\,(\Sigma)$.

Sean A y B dos AFR tales que $L_1 = L(A)$ y $L_2 = L(B)$. Ya se demostró en la proposición 2.2.10 que $L_1 + L_2$ es regular. Cuesta poco sin embargo crear un autómata según el esquema de la figura 3.3. En él, el estado α es el nuevo estado inicial y mediante dos transiciones vacías vamos a los estados iniciales de A y B. Se comprueba fácilmente que este autómata acepta el lenguaje $L_1 + L_2$.

Basándonos en la figura 3.4, construimos un autómata que acepte L_1L_2. Está claro que si $w \in L_1L_2$, donde $w = x \cdot y$, $x \in L_1$ y $y \in L_2$. La palabra x nos lleva a un estado final en A del cual pasamos con una transición vacía al autómata B en donde la palabra y nos lleva a un estado final.

Finalmente, no es difícil comprobar que el autómata de la figura 3.5 reconoce el lenguaje L_1^*. ■

Para probar la otra inclusión del teorema de Kleene necesitamos probar que el lenguaje aceptado por un autómata finito se puede describir mediante una expresión racional. Para ello vamos a introducir sistemas de ecuaciones con lenguajes. En concreto vamos a considerar sistemas de ecuaciones lineales en $\mathcal{P}(\Sigma^*)$ del tipo siguiente

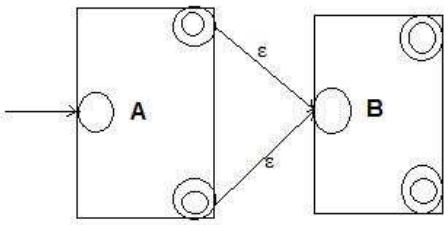


Figura 3.4: Producto de lenguajes

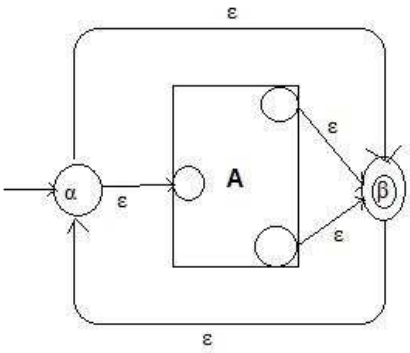


Figura 3.5: estrella de kleene

$$X_k = \sum_{j=1}^{n} X_j E_{jk} + T_k, \quad 1 \leq k \leq n \tag{3.1}$$

donde $X_k, E_{jk}, T_k \in \mathcal{P}(\Sigma^*)$. Las X_k son las **incógnitas**, los E_{jk} los **coeficientes** y T_k los **términos independientes**. Lo podemos representar matricialmente de la siguiente forma:

$$\begin{pmatrix} X_1 & \dots & X_n \end{pmatrix} = \begin{pmatrix} X_1 & \dots & X_n \end{pmatrix} \begin{pmatrix} E_{11} & E_{12} & \dots & E_{1n} \\ E_{21} & E_{22} & \dots & E_{2n} \\ \vdots & \vdots & & \vdots \\ E_{n1} & E_{n2} & \dots & E_{nn} \end{pmatrix} + \begin{pmatrix} T_1 & \dots & T_n \end{pmatrix}$$

Veremos que este tipo de sistemas tienen casi siempre solución única. Empecemos por el sistema formado por una sola ecuación.

Lemma 3.2.7 (Arden)**.** *Sea $X = XE + T$ una ecuación en $\mathcal{P}(\Sigma^*)$.*

(a) $X = TE^*$ es una solución.

(b) Si Y es una solución, entonces $TE^* \subseteq Y$.

(c) Si $\epsilon \notin E$, entonces TE^* es la única solución.

Demostración. (a) Utilizando las propiedades de la proposición 3.2.2 tenemos que

$$XE + T = TE^*E + T = T(EE^* + \epsilon) = T(E^+ + \epsilon) = TE^*.$$

(b) Sea Y una solución. Entonces se cumple que $T \subseteq Y$ e $YE \subseteq Y$, de donde $TE \subseteq Y$ y, por tanto, $TE^2 \subseteq YE \subseteq Y$. Por un argumento inductivo obtenemos $TE^n \subseteq Y \ \forall n \in \mathbb{N}$. Por lo tanto, $TE^* \subseteq Y$.

(c) Sea W una solución de la ecuación. Por el apartado anterior $TE^* \subseteq W$. Sea $z \in W \setminus TE^*$ con $|z|$ mínimo. Como $z \notin TE^*$ tenemos que $z \notin T$. Luego $z \in WE$. Pongamos $z = wc$ con $c \in E$ y $w \in W$. Dado que $\epsilon \notin E$, se tiene que $|w| < |z|$. Por último, veamos que $w \notin TE^*$, lo cual será contradictorio con la minimalidad de $|z|$. Si $w \in TE^*$, entonces $z = wc \in TE^*E \subseteq TE^*$ lo cual es contradictorio. ∎

Ejemplo 3.2.8. Si tenemos la ecuación $X = X\epsilon + T$, está claro que cualquier conjunto L que contenga a T será solución.

Veamos ahora que los sistemas 3.1 siempre tienen solución.

Proposición 3.2.9. *El sistema de ecuaciones 3.1 siempre tiene solución. Si además $E_{ij}, T_j \in Rat(\Sigma^*)$, entonces la solución también es racional.*

DEMOSTRACIÓN. Procedamos por inducción sobre el número de ecuaciones. Si $n = 1$ el sistema es $X_1 = X_1E_{11} + T_1$. Por el lema anterior, $X_1 = T_1E_{11}^*$ que es racional si T_1 y E_{11} son racionales.

Supongamos $n > 1$. La última ecuación es

$$X_nE_{nn} + (C + T_n)$$

donde $C = X_1E_{1n} + \ldots + X_{n-1}E_{n-1n}$. Por el lema de Arden, $X_n = (C + T_n)E_{nn}^*$ y sustituyendo en las ecuaciones anteriores se tiene un sistema de n-1 ecuaciones

$$X_j = \sum_{i=1}^{n-1} X_i(E_{ij} + E_{in}E_{nn}^*E_{nj}) \ + (T_nE_{nn}^*E_{nj} + T_j) \quad j = 1, ..., n-1$$

Este sistema por inducción tiene solución y ésta es racional. Sustituyendo en C tenemos X_n y también es racional. ■

A continuación vamos a ver que si los coeficientes E_{ij} no contienen la palabra vacía, entonces la solución es única. Para ello necesitamos introducir las funciones características.

Dado un $L \in \mathcal{P}(\Sigma^*)$, definimos la **función característica** de L como la función

$$\hat{L} : \Sigma^* \longrightarrow \{0, 1\} = \mathbb{B}$$

definida como

$$\bar{L}(s) = \begin{cases} 0 & \text{si } s \notin L \\ 1 & \text{si } s \in L \end{cases}$$

Está claro que hay una correspondencia biyectiva entre $\mathcal{P}(\Sigma^*)$ y el conjunto de aplicaciones de Σ^* en $\mathbb{B}$. Consideremos en $\mathbb{B}$ las operaciones booleanas (que hacen de $\mathbb{B}$ una álgebra de Boole):

$+$	0	1
0	0	1
1	1	1

$\cdot$	0	1
0	0	0
1	0	1

En el conjunto de aplicaciones de Σ^* en $\mathbb{B}$ definimos las operaciones

$$\begin{cases} (\hat{L} + \hat{L'})(s) = \hat{L}(s) + \hat{L'}(s) \\ (\hat{L} \cdot \hat{L'})(s) = \sum_{tu=s} \hat{L}(t)\hat{L'}(u) \end{cases}$$

Entonces no es difícil comprobar que $\widehat{L + L'} = \hat{L} + \hat{L'}$ y $\widehat{L \cdot L'} = \hat{L} \cdot \hat{L'}$.

Proposición 3.2.10. *Si $\epsilon \notin E_{ij}, \forall i,j$, entonces la solución del sistema 3.1 es única.*

DEMOSTRACIÓN. Sean $(X_1,\ldots,X_n)$ y $(Y_1,\ldots,Y_n)$ dos soluciones del sistema. Vamos a demostrar que $\hat{X}_j(s) = \hat{Y}_j(s) \ \ \forall s \in \Sigma^*$.

Procederemos por inducción sobre $|s|$. Si $|s| = 0$, entonces $s = \epsilon$ y, por hipótesis, $\hat{E}_{ij}(\epsilon) = 0$. Entonces tenemos

$$\hat{X}_j(\epsilon) = \sum_{i=1}^{n} \widehat{X_i E_{ij}}(\epsilon) + \hat{T}_j(\epsilon) = \hat{T}_j(\epsilon)$$

y

$$\hat{Y}_j(\epsilon) = \sum_{i=1}^{n} \widehat{Y_i E_{ij}}(\epsilon) \ + \ \hat{T}_j(\epsilon) = \hat{T}_j(\epsilon).$$

Supongamos$|s| > 0$. Entonces

$$\hat{X}_j(s) = \sum_{i=1}^{n} \widehat{X_i E_{ij}}(s) \ + \hat{T}_j(s) = \sum_{i=1}^{n} \sum_{uv=s} \hat{X}_i(u)\hat{E}_{ij}(v) + \hat{T}_j(s)$$

Dado que $\hat{E}_{ij}(\epsilon) = 0$, tenemos que

$$\hat{X}_j(s) = \sum_{i=1}^{n} \sum_{\substack{uv=s \\ |u|<|s|}} \hat{X}_i(s)\hat{E}_{ij}(v) \ \ + \hat{T}_j(s)$$

Por hipótesis de inducción $\hat{X}_j(u) = \hat{Y}_j(u)$ y por tanto obtenemos que $\hat{X}_j(s) = \hat{Y}_j(s)$. ∎

Ejemplo 3.2.11. Consideramos el siguiente sistema

$$\begin{cases} X = Xa + Yb \\ Y = Xa + Yb + \epsilon \end{cases}$$

Para resolverlo utilizamos la última ecuación $Y = Yb + (Xa + \epsilon)$. Según el lema de Arden

$$Y = (Xa + \epsilon)b^*. \tag{3.2}$$

Sustituyendo y reordenando en la primera ecuación

$$X = Xa + (Xa + \epsilon)b^*b = Xa + (Xa + \epsilon)b^+ = X(a + ab^+) + b^+ = Xab^* + b^+$$

y otra lema por el lema de Arden

$$X = b^+(ab^*)^*.$$

Finalmente, sustituyendo en 3.2 tenemos $Y = Yb + (b^+(ab^*)^*a + \epsilon)$, de donde

$$Y = (b^+(ab^*)^*a + \epsilon)b^* = b^+(ab^*)^*ab^* + b^* = b^+(ab^*)^+ + b^*.$$

Teorema 3.2.12 (Kleene, 1956)**.** *Sea Σ un conjunto finito, entonces $Reg(\Sigma) = Rat(\Sigma^*)$.*

DEMOSTRACIÓN. Hemos visto en la proposición 3.2.6 que $Rat(\Sigma^*) \subseteq \mathrm{Reg}(\Sigma)$. Para demostrar la otra inclusión, vamos a ver que lenguaje reconocido por un autómata finito es la solución de un sistema de ecuaciones como los considerados anteriormente. Sea $A = (Q, \Sigma, q_0, \delta, F)$ un autómata, que podemos suponer accesible, que reconoce el lenguaje $L(A)$. Para simplificar la notación, supongamos que $Q = \{q_0 = 1, \ldots, n\}$ y definamos

$$X_j = \{u \in \Sigma^* | 1u = j\} \qquad E_{jk} = \{a \in \Sigma \mid ja = k\}$$

Diremos que X_j es el lenguaje aceptado por el estado j. Tenemos entonces la siguiente igualdad

$$L(A) = \sum_{j \in F} X_j.$$

Veamos que se satisface el siguiente sistema de ecuaciones

$$X_k = \sum_{j=1}^{n} X_j E_{jk} + T_k \qquad 1 \leq k \leq n$$

con $T_j = \emptyset$ si $j \neq 1$ y $T_1 = \epsilon$.

Una inclusión es sencilla, ya que $X_j E_{jk} \subseteq X_k$. Veamos la otra inclusión.

Sea $u \in X_k$, entonces $1u = k$. Si $u = \epsilon$, entonces $k = 1$ y $u \in T_1$. Supongamos $u \neq \epsilon$. Sea $u = s_1 \cdots s_t$, $s_i \in \Sigma$ y pongamos $v = s_1 \cdots s_{t-1}$. Supongamos que $1v = j$. Entonces $v \in X_j$ y $s_t \in E_{jk}$ y, por lo tanto, $u \in \sum_{i=1}^{n} X_j E_{jk} + T_k$. ∎

Corolario 3.2.13. *Sea Σ un conjunto finto, entonces $Reg(\Sigma) = Rec(\Sigma^*) = Rat(\Sigma^*)$.*

DEMOSTRACIÓN. Teorema 3.2.6 y teorema 2.2.11. ∎

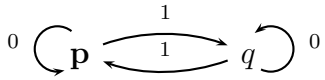
Ejemplo 3.2.14. Podemos preguntarnos cómo se comportan los lenguajes regulares bajo un homomorfismo de semigrupos. En concreto, sean Σ y Ω dos conjuntos finitos y $f : \Sigma^* \to \Omega^*$ un homomorfismo de monoides. Sea $L \subseteq \Sigma^*$ un lenguaje regular. Por el corolario anterior, L es racional y entonces $f(L)$ también es racional y, por tanto, regular. En efecto, pues basta observar que si $A, B \subseteq \Sigma^*$, se tiene

$$f(AB) = f(A)f(B), \quad f(A + B) = f(A) + f(B), \quad f(A^+) = f(A)^+.$$

Para las antiimágenes, el resultado también es cierto. Se puede construir directamente una autómata finito que reconozca $f^{-1}(K)$ con K regular, o bien se puede utilizar el teorema de Myhill-Nerode (véase por ejemplo [1, pág 93]).

Vamos a utilizar lo anterior para ver algo que quedó pendiente en el ejemplo 2.1.10. Dado el alfabeto $\Sigma = \{a, b\}$, sea el lenguaje $M = a^+L + a^*$, con $L \subseteq b^*$ no regular. Vamos a probar que M no es regular. Para ello definimos $f : (a+b)^* \to b^*$ por $f(a) = \epsilon$ y $f(b) = b$. Está claro que $f(M) = L$ y si L no es regular, entonces M tampoco.

Ejemplo 3.2.15. Consideremos el autómata



El estado p es el estado inicial y final. Calculamos E y T según la demostración del teorema anterior,

$$E = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad T = \begin{pmatrix} \epsilon & \emptyset \end{pmatrix}$$

Tenemos el siguiente sistema:

$$\begin{cases} X_p = X_p 0 + X_q 1 + \epsilon \\ X_q = X_p 1 + X_q 0 \end{cases}$$

De la última ecuación obtenemos $X_q = X_p 10^*$ y sustituyendo en la primera

$$X_p = X_p 0 + X_p 10^* + \epsilon = X_p(0 + 10^*) + \epsilon.$$

De donde finalmente obtenemos

$$L = (0 + 10^*1)^*$$

Bibliografía

[1] Mark V. Lawson, *Finite Automata*, Chapman & Hall (2004).

[2] De Luca, A. and Varrichio, S., *Finiteness and regularity in semigroups and formal languages*, Springer-Verlag (1999).

[3] Grillet, P.A., *Semigroups: an introduction to the structure theory*, Marcel Dekker (1995).

[4] Hopcroft, J.E. and Ullman, J.D., *Introduction to automata theory, languages and conputation*, Addison-Wesley (1979).