Lenguajes y álgebra de eventos regulares

Pablo Gerardo Padilla Beltrán
Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
tyomero@gmail.com

Septiembre 20, 2006

Resumen

Este reporte contiene información introductoria a la teoría de los lenguajes formales, autómatas y algebra de eventos regulares. Temas que fuerón mencionados en algunas de las clases del verano.

La sección 2 introduce los conceptos más básicos de la teoría de lenguajes formales, los cuales son necesarios para entender la sección 3, ya que en todo momento se hace referencia a operaciones sobre lenguajes, que posteriormente son llamados eventos.

La última sección es un resumen de la demostración propuesta por John H. Conway [JC71] para el torema de Kleene acerca de la representabilidad de eventos. Durante la cual desarrolló un metódo para pasar de máquinas de estados finitos a expresiones regulares.

Contenido

1 Introducción

Aquellos que esten interesados en la teoría de los lenguajes formales y autómatas, encontrarán una explicación breve acerca de los conceptos básicos de las mismas. En las secciones iniciales se tratan los aspectos fundamentales de dichas teorías, pasando después a explicar brevemente las características de algunos tipos básicos de autómata y sus relaciones con los diferentes tipos de lenguajes y sus respectivas gramáticas.

A partir de la sección 3 se examinan conceptos relacionados con el álgebra de eventos regulares. Para una mejor comprensión de el álgebra regular, se analizan primero, algunos conceptos de la teoría de los experimentos de Moore, ésto con la finalidad de acostumbrarse a la conceptualización de Moore acerca de una máquina de estados finitos. A partir de la sección 3.2 podemos aventurarnos en el tema del álgebra regular, la cual ha sido mayormente examinada por John H. Conway en [JC71, cap. 3], tomando como punto de partida la teoría de eventos regulares, definida por Stephen Cole Kleene.

2 Alfabetos y lenguajes

2.1 Alfabetos, cadenas y lenguajes

Definición 2.1. *Un alfabeto es un conjunto finito no vacío de símbolos y se denota como* Σ.

La pertenencia de un símbolo σ a un alfabeto Σ se denota como $\sigma \in \Sigma$.

Ejemplo: Podemos representar el alfabeto de las letras minúsculas que utiliza el idioma español, el cual contiene los 27 símbolos siguientes:

$$\Sigma = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, \tilde{\text{n}}, o, p, q, r, s, t, u, v, w, x, y, z\},$$

y sabemos que la letra a pertenece a este alfabeto, lo cual denotaremos como $a \in \Sigma$.

Ya sabemos que los alfabetos son conjuntos, por lo que, todas las operaciones de conjuntos se pueden aplicar a los alfabetos también. Sean $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ alfabetos, y ya que los alfabetos son conjuntos finitos, no vacíos, la unión de un número finito de ellos $\bigcup_{i=1}^{n} \Sigma_i$ resulta en un conjunto no vacío y finito, esto es, si $\Sigma_i \neq \emptyset$ y $\Sigma_{i+1} \neq \emptyset$, entonces $\Sigma_i \cup \Sigma_{i+1} \neq \emptyset$.

La unión de un número arbitrario finito de alfabetos resultará en un conjunto finito y no vacío, es más, si Σ_1 y Σ_2, son conjuntos no vacíos, entoces $\Sigma_1 - \Sigma_2$, $\Sigma_2 - \Sigma_1$ y $\Sigma_1 \cap \Sigma_2$ son conjuntos finitos, no vacíos, y por lo tanto serán considerados alfabetos válidos.

Definición 2.2. *Una cadena o palabra es una secuencia finita de símbolos que pertenecen a un alfabeto y comunmente se denota con la letra* w. *La cadena vacía se denota como* ε *y es una secuencia vacía de símbolos tomados de cualquier alfabeto* Σ.

Sí el alfabeto es el español, algunas cadenas pueden ser *yomero*, *tumero* y *malnacido*. Dada la definición anterior, cualquier palabra que contenga los símbolos del alfabeto es una cadena válida, sin importar si esta tiene o no significado alguno.

Si w es cualquier cadena, su longitud se denota como $|w|$, la longitud de una cadena es el número de símbolos que contiene, por ejemplo, si tenemos la cadena $w = malnacido$ sobre el alfabeto español, $|w| = 9$. La cadena vacía ε no tiene símbolos, por lo que $|\varepsilon| = 0$

Definición 2.3. *Un lenguaje* L *es un conjunto de cadenas sobre un alfabeto* Σ *definido, éstas pueden ser cualquier cadena* w, *que cumpla con lo siguente,* w *esta formada por los símbolos* $\sigma_1\sigma_2 \ldots \sigma_k$ *donde* $\sigma_k \in \Sigma\ \forall k$.

El lenguaje vacío es aquel que no contiene cadenas y no es lo mismo que el lenguaje formado por la cadena vacía $\{\varepsilon\}$, éste lenguaje se denota de la misma manera que el conjunto vacío, $\emptyset$.

Sí se tiene una cadena w sobre un alfabeto Σ y L es el lenguaje compuesto por algunas de las cadenas sobre el alfabeto Σ y $w \in L$, entonces diremos que w es un miembro de L.

Definición 2.4. *Un lenguaje universal sobre algún alfabeto* Σ, *o cerradura de* Σ, *es el lenguaje que contiene todas las cadenas que es posible formar con los símbolos de* Σ *y se denota como* Σ^*.

Ejemplo: Sea $\Sigma = \{a\}$, entonces $\Sigma^* = \{\varepsilon, a, aa, aaa, \ldots\}$.

Podemos observar que para cualquier alfabeto Σ, Σ^* es infinito, ya que los alfabetos son conjuntos no vacíos.

2.2 Operaciones con cadenas

Ahora describiremos algunas de las características de las cadenas y operaciones básicas que se pueden realizar con ellas.

Si w_1 y w_2 son cadenas, la *concatenación* de éstas dos cadenas resulta en la cadena que se obtiene al agregar la segunda al final de la primera, es decir, si tenemos $w_1 = mesa$ y $w_2 = banco$, la concatenación de estas dos cadenas es *mesabanco* y se denota $w_1 \cdot w_2$ o $w_1 w_2$, por lo que podemos observar que $|w_1 + w_2| = |w_1| + |w_2|$. La concatenación de cualquier cadena w_1 con la cadena vacía ε deja intacta a la cadena w_1. La igualdad entre cadenas se denota con el signo '=' y se dá cuando dos o más cadenas tienen exactamente los mismos símbolos en la misma posición y tienen la misma longitud.

Ejemplo: Si $w_1 = luz$ y $w_2 = luz$, entonces $w_1 = w_2$.

La *potencia* de una cadena sobre un alfabeto quiere decir que tomamos toda la cadena como una unidad atómica, es decir, si $w = abc$, entonces $w^2 = ww, w^3 = www$ y así sucecivamente. Lo anterior lo podemos simplificar con la siguiente definición.

$$w^n = \begin{cases} \varepsilon, & \text{si } n = 0 \\ ww^{n-1} & \text{si } n > 0 \end{cases}$$

Por lo tanto si $w = ab$ sobre el alfabeto $\Sigma = \{a, b\}$, tenemos que

$$\begin{aligned} w^0 &= \varepsilon \\ w^1 &= ab \\ w^2 &= abab \\ w^3 &= ababab, \end{aligned}$$

y podemos continuar hasta la i-ésima *potencia* de w, que denotaremos como w^i.

Ahora trataremos con el *sufijo* y el *prefijo* de una cadena, si tenemos una cadena $w = xy$, donde x y y también son cadenas, entonces x es el *prefijo* de w y y es el sufijo, hay que recordar que la cadena vacía puede ser el prefijo de cualquier cadena, además, si tenemos que $c = xy$, donde x es el prefijo de c y $y = \varepsilon$, entonces resulta que $x = c$, lo cual indica que toda cadena es prefijo de si misma.

Una cadena c es una *subcadena* o *subpalabra* de otra cadena w, si exiten cadenas x y y para las cuales $w = xcy$.

La *inversa* o *transpuesta* de una cadena w se denota como w^I y quiere decir que, si se tiene $w = taza$, entonces $w^I = azat$. Más generalmente podemos decir que

$$w^I = \begin{cases} w, & \text{si } w = \varepsilon \\ y^I a, & \text{si } w = ay \text{ por lo tanto } a \in \Sigma \text{ y } y \in \Sigma^* \end{cases}$$

Para ver como funciona la definición anterior utilizaremos un ejemplo, si se tiene $w = tabla$, al aplicar la definición, se logra lo siguiente:

$$\begin{aligned} w^I = (tabla)^I &= (abla)^I t \\ &= (bla)^I at \\ &= (la)^I bat \\ &= (a)^I lbat \\ &= (\varepsilon)^I albat \\ &= \varepsilon albat \\ &= albat \end{aligned}$$

La inversa de una cadena tiene ciertas características, con la concatenación de cadenas, por ejemplo, si se tienen cadenas ab y cd que al concatenarse forman $abcd$, sabemos que $(abcd)^I = dcba$, de lo cual podemos observar que $dcba = (cd)^I(ab)^I$. Por lo tanto, si g y h son cadenas y si $x = gh$, entoces $x^I = h^I g^I$.

La inversa se anula a si misma, es decir, si a una cadena w se le aplica la inversa dos veces seguidas, el resultado será w, como si no se hubiera aplicado ni una vez. Más generalmente, $(w^I)^I = w$.

Ejemplo:

$$\begin{aligned}((abcd)^I)^I &= (dcba)^I \\ &= abcd\end{aligned}$$

2.3 Operaciones con lenguajes

El concepto de *concatenación* se puede extender a los lenguajes. Se define la concatenación de lenguajes como sigue:

$$A \cdot B = \{a \cdot b |\ a\ \in A \text{ y } b\ \in\ B\}$$

El lenguaje que resulta de la concatenación de A y B esta formado por la concatenación de todas la cadenas de A con todas la cadenas de B.

Ejemplo: Si $A = \{hola,\ adios\}$ y $B = \{casa\}$, entonces, $A \cdot B = \{holacasa,\ adioscasa\}$.

La concatenación de lenguajes se puede realizar aún si los lenguajes no estan contruídos sobre el mismo alfabeto, en tal caso la concatenación nos lleva a que, si A y B, son lenguajes sobre Σ_1 y Σ_2, entonces el lenguaje resultante será un lenguaje sobre $\Sigma_1 \cup \Sigma_2$.

La cadena vacía se comporta como la identidad en cuanto a la concatenación de lenguajes se trata, ya que si tenemos $A \cdot \{\varepsilon\} = \{\varepsilon\} \cdot A = A$.

La definición de *potencia*, también puede extenderse a los lenguajes de la misma manera

$$A^n = \begin{cases} \{\varepsilon\}, & \text{si } n = 0 \\ A \cdot A^{n-1}, & \text{si } n \geq 0 \end{cases}$$

Por lo tanto, si $A = \{ab\}$ sobre un algún alfabeto, se tiene que

$$\begin{aligned}A^0 &= \{\varepsilon\} \\ A^1 &= A = \{ab\} \\ A^2 &= A \cdot A^1 = \{abab\} \\ A^3 &= A \cdot A^2 = \{ababab\}\end{aligned}$$

Hay que destacar que de la anterior definición se tiene que $\emptyset^0 = \{\varepsilon\}$.

Dado que los lenguajes son conjuntos de cadenas, las operaciones, intersección, unión y sublenguaje se difinen como sigue:

Sean A y B lenguajes sobre el alfabeto Σ, la *unión* se denota como $A \cup B$ y quiere decir que el lenguaje resultante esta formado por todas la palabras que se encuentren en al menos uno de los dos lenguajes, más generalmente:

$$A \cup B = \{x \mid x\ \in\ A \text{ o } x\ \in\ B\}.$$

La *intersección* de los lenguajes A y B es un lenguaje formado por todas las cadenas que se encuentran tanto en A como en B, esto es:

$$A \cap B = \{x \mid x\ \in\ A \text{ y } x\ \in\ B\}$$

Con un ejemplo se prodrá ilustrar mejor las dos definiciones anteriores.

Ejemplo: Sean $A = \{\varepsilon, a, b, c, d, e\}$ y $B = \{\varepsilon, 0, 1, 2, 3, 4\}$,

$$A \cup B = \{\varepsilon, a, b, c, d, e, f, 0, 1, 2, 3, 4\} \text{ y}$$

$$A \cap B = \{\varepsilon\}.$$

Ahora hay que definir el concepto de sublenguaje, recordando la teoría de conjuntos sabemos que un conjunto W es un subconjunto de U, si U contiene a todos los elementos de W y se denota como $W \subseteq U$, y se lee, *W es un subconjunto de U*. Esta definición se puede mudar perfectamente a la teoría de lenguajes, diciendo que si A y B son lenguajes, entonces B es un sublenguaje de A, si A contiene todas las cadenas de B y se denota $B \subseteq A$, y se lee *B es un sublenguaje de A*.

Sea L cualquier lenguaje sobre Σ, entonces $L \subseteq \Sigma^*$, ya que Σ^* contiene todas las cadenas que son posibles de generar con el alfabeto Σ.

La igualdad de lenguajes cumple con las mismas caracteríscas que la igualdad entre conjuntos, sean A y B lenguajes, son iguales, sólo si, contienen exactamente las mismas cadenas. También hereda sus propiedades de la teoría de conjuntos, tales como son los siguientes teoremas:

Teorema 2.3.1. *Sean A y B, lenguajes sobre Σ, $A = B$, solo si $A \subseteq B$ y $B \subseteq A$.*

El teorema 2.3.1 sirve para demostrar la igualdad entre lenguajes y se utiliza para demostrar que la concatenación es distributiva con respecto a la unión de lenguajes.

Demostración. Supongamos que $A = B$, entónces tenemos que probar que $A \subseteq B$ y $B \subseteq A$, para ello digamos que $x \in A$. Como B contiene las mismas cadenas que A, diremos que $x \in B$, de lo que se deduce que $A \subseteq B$. De la misma forma, si $x \in B$, entónces $x \in A$ ya que los dos contienen las mismas cadenas, de lo anterior tenemos que $B \subseteq A$, lo cual no lleva a que $A \subseteq B$ y $B \subseteq A$. Esto significa que las cadenas que estan en B, están también en A y viceversa, por lo que $A = B$, con lo que se demuestra la igualdad. □

Teorema 2.3.2. *Dados los lenguajes A, B y C, sobre un alfabeto Σ, se cumple que:*

1. $A \cdot (B \cup C) = A \cdot B \cup A \cdot C$
2. $(B \cup C) \cdot A = B \cdot A \cup C \cdot A$

Demostración. Para demostrar la primera parte del teorema, probaremos primero que $A \cdot (B \cup C) \subseteq A \cdot B \cup A \cdot C$. Supongamos que $x \in A \cdot (B \cup C)$, y que $x = w \cdot y$, donde $w \in A$ y $y \in B \cup C$. Sí $y \in B$, tenemos que $x = w \cdot y \in A \cdot B$ y por lo tanto $x \in A \cdot B \cup A \cdot C$. Si $y \in C$, tenemos que $x \in A \cdot C$ y de nuevo $x \in A \cdot B \cup A \cdot C$. Sin importar a que lenguaje pertenesca y, se deduce que

$$A \cdot (B \cup C) \subseteq A \cdot B \cup A \cdot C$$

Ahora para probar $A \cdot B \cup A \cdot C \subseteq A \cdot (B \cup C)$ suponemos que $x \in A \cdot B \cup A \cdot C$ de modo que $x \in A \cdot B$ o $x \in A \cdot C$. Si $x \in A \cdot B$ y $x = u \cdot v$ donde $u \in A$ y $v \in B$, tenemos que $v \in B \cup C$, y ya que $u \in A$, tenemos que $x \in A \cdot (B \cup C)$. Por otro lado si $x \in A \cdot C$ y si $x = w \cdot y$, tenemos que $w \in A$ y $y \in C$, por lo tanto $y \in B \cup C$ y ya que $w \in A$, tenemos que $x \in A \cdot (B \cup C)$. De lo anterior se deduce que $A \cdot B \cup A \cdot C \subseteq A \cdot (B \cup C)$. Utilizando el teorema 2.3.1 se obtiene que $A \cdot B \cup A \cdot C = A \cdot (B \cup C)$, lo que demuestra la igualdad. □

De forma muy similar se demuestra la segunda parte del teorema 2.3.2. Así que no aparecerá la demostración en este documento.

A diferencia de la unión, la concatenación no es distributiva con respecto a la intersección de lenguajes, para esto, hay que proponer que, si $A = \{a, \varepsilon\}$, $B = \{\varepsilon\}$ y $C = \{a\}$, entoces $A \cdot B = \{a, \varepsilon\}$ y $A \cdot C = \{a^2, a\}$, por lo que $A \cdot B \cap A \cdot C = \{a\}$. Pero tenemos que $B \cap C = \emptyset$, entonces $A \cdot (B \cap C) = \emptyset$, por lo tanto:

$$A \cdot B \cap A \cdot C = \{a\} \neq \emptyset = A \cdot (B \cap C)$$

Ahora veremos dos conceptos más, el primero es el de *cerradura de Kleene* o *cerradura estrella*, élla esta definida como la unión de 0 o más potencias de un lenguaje A sobre un alfabeto Σ, más precisamente, la *cerradura de Kleene* es realizar 0 o más concatenaciones del lenguaje A con él mismo, y se denota $A^* = \cup_{n=0}^{\infty} A^n$, lo que resulta en un lenguaje que contiene todas las cadenas que son posibles de formar sobre Σ. También tenemos a la *cerradura positiva*, que es la unión de una o más potencias de A en Σ, resultando en un lenguaje que contiene, todas las cadenas excepto la cadena vacía ε, y se denota $A^+ = \cup_{n=1}^{\infty} A^n$.

Un factor importante que debe recordarse es que la diferencia entre estos dos tipos de cerradura es, que en la *cerradura de Kleene* se realiza con 0 o más concatenaciones, en cambio la *cerraduta positiva* se realiza con 1 o más concatenaciones.

Ejemplo: Si Σ es el alfabeto español y $A = \{a\}$ sobre Σ, tendremos que $A^* = \cup_{n=0}^{\infty} A^n = \{\varepsilon, a, a^2, \ldots\}$, ya que $A^0 = \{\varepsilon\}$ y $A^+ = \cup_{n=1}^{\infty} A = \{a, a^2, a^3, \ldots\}$.

La definición de la cerradura de Kleene es igual a la de el lenguaje universal, mencionada en la sección 2.4 en la página 2. Sea Σ un alfabeto, $\Sigma^* = \cup_{i=0}^{\infty} \Sigma$ es la concatenación de 0 o más símbolos de Σ, que son las cadenas que conforman el lenguaje universal que también se denota Σ^*, de aquí que todo lenguaje sobre Σ es un sublenguaje de Σ^*.

La diferencia entre lenguajes sigue las mismas reglas que para la diferencia en conjuntos, es decir, si A y B son lenguajes sobre Σ, entonces $A - B = \{x \mid x \in A \text{ y } x \notin B\}$, que resulta en un lenguaje que contiene todas las cadenas de A, que no estan en B.

Al igual que con conjutos se puede definir *el complento de un lenguaje*. Sea A un lenguaje sobre Σ su complemento es

$$\overline{A} = \Sigma^* - A$$

se puede que ver la definición se plantea de la misma manera que con conjuntos, donde el lenguaje complemento contiene todas las cadenas del lenguaje universal Σ^*, que no estan en A.

El *inverso* de un lenguaje se denota como A^I y su efecto sobre el lenguaje es que todas las cadenas del lenguaje se invierten, esto es, si A es un lenguaje, su inverso es $A^I = \{x^I \mid x \in A\}$.

Ejemplo: Si $A = \{hola, raro\}$, entonces $A^I = \{aloh, orar\}$.

El inverso de un lenguaje se anula a sí mismo. Al igual que con las cadenas, el inverso del inverso de un lenguaje, deja el lenguaje original intacto, esto es, $(A^I)^I = A$.

El uso de el inverso de un lenguaje es bueno para casí todas las operaciones sobre lenguajes, pero en el caso de la concatenación, no solo invierte las palabras concatenadas de los lenguajes, sino que también cambia el ordan de la concatenación de los lenguajes, $(A \cdot B)^I = B^I A^I$.

2.4 Expresiones Regulares

Las expresiones regulares son usadas para definir patrones, los cuales pueden encontrarse en cadenas o palabras de algún lenguaje, pero no estan restringidas solo a este contexto, pueden ser también cualquier patrón que tenga cierta recurrencia en cualquier otro contexto. Son utilizadas en muchas de las herramientas para análisis léxico, donde se trabaja directamente con cadenas de texto para encontrar los patrones que existen en el lenguaje.

Las expresiones regulares se clasifican de la siguiente manera:

Sea Σ un alfabeto, y w una palabra, tal que $w \in \Sigma$, entonces:

- w por si sola es una expresión regular. Representa el conjunto que contiene todas las cadenas de un solo símbolo.
- Si e_1 y e_2 son ambas expresiones regulares, $e_1 \cdot e_2$ es una expresión regular.
- Si $e_1, e_2, \ldots, e_{n-1}, e_n$, son expresiones regulares, la unión $(e_1 \cup e_2 \cup \ldots \cup e_{n-1} \cup e_n)$ resulta en una expresión regular.
- Si e es una expresión regular, entonces e^* también lo es.
- Las expresiones regulares son las mencionadas antes, ninguna otra expresión es regular.

Podemos simplificar la notación de un lenguaje usando las expresiones regulares ya que funcionan como una plantilla para indicar la forma que deben tener todas las cadenas que forman parte de algún lenguaje.

$a \cup b$	denota el lenguaje	$\{a\} \cup \{b\} = \{a, b\}$
$a \cdot b$	denota el lenguaje	$\{a\} \cdot \{b\}$
a^*	denota el lenguaje	$\{a\}^*$
a^+	denota el lenguaje	$\{a\}^+$

Cualquier tipo de lenguaje se puede describir por medio de expresiones regulares. Existen ciertas equivalencias entre distintos tipos de máquinas de estados finitos, de las cuales se dará una descripción breve más adelante.

2.5 Autómata Finito

Recordando lo visto en la sección 2.4, una expresión regular es una forma de describir un lenguaje, pero, ¿Cómo haríamos para saber si una cadena pertenece o no al lenguaje?

Una forma de hacerlo es construyendo un diagrama que contenga la información necesaria para analizar la cadena, este diagrama tendrá forma de grafo dirigido[1]. La cadena será leída de izquierda a derecha.

Para dar a conocer que la cadena es aceptada, se debe establecer un estado de aceptación y también se deben establecer estados de no aceptación para terminar el análisis de las cadenas que no pertenescan al lenguaje. Los estados, tanto de aceptación como los que no lo son, serán representados por medio de nodos en el grafo, los cuales es conveniente etiquetar de alguna forma, así que definiremos $Q = \{q_1, q_2, \ldots, q_n\}$ como el conjunto finito de todos los estados posibles, q_1 será el estado inicial y $F \subseteq Q$ será el conjunto finito de todos los estados finales, es decir, estados de aceptación.

Tomaremos a la cadena que será analizada como nuestro alfabeto de entrada y lo denotaremos como Σ. Cada arista en el grafo es etiquetada con un símbolo de Σ que indica una transición de algún estado a otro en el sentido marcado por la flecha en algúno de los extremos de cada arista. Al analizar un carácter es posible permanecer en el mismo estado después de la transición. Los estados de aceptación deben de distinguirse de los demás, por lo que serán encerrados en un círculo.

Analizar una cadena es recorrer los caminos del grafo; dependiendo del carácter que esta siendo análizado se elige un camino a recorrer. Cuando se acepta una cadena como miembro del lenguaje se llega a un estado de aceptación (encerrados en un círculo) por lo que, todos los caminos que lleven a un estado de aceptación construyen palabras pertenecientes a el lenguaje.

Ejemplo: Si tenemos, el alfabeto $\Sigma = \{a, b\}$ y el lenguaje descrito por la expresión regular $(ab)^* = \{\varepsilon, ab, (ab)^2 \ldots, (ab)^n\}$, para demostrar que la cadena $c = ab$ pertenece a este lenguaje construimos el diagrama de la figura 1.

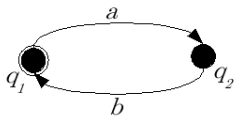

Figura 1: Gráfo para analizar el lenguaje descrito por la expresión regular $(ab)^*$

También se puede describir la forma en que la cadena es procesada por medio de un tabla, la cual contiene la información acerca de las transiciones que se realizan al recorrer la cadena. En dicha tabla tendremos relacionadas parejas de la forma (q_i, α), donde $q_i \in S$, $i = (1, 2, \ldots)$ y α es un carácter de la cadena que se esta analizando. A partir de un estado y un carácter podremos decir cual será el camino que debe ser recorrido en el grafo, el cual nos llevará hacía algún estado en Q, por lo que podemos definir una función $\delta(q_i, \alpha)$, que depende de dos variables, el estado actual y el carácter analizado, en otras palabras la función tomará pares de la forma $Q \times \Sigma$, para definir a que estado se llegará, entonces podemos definir más generalmene la función de transición de estados como sigue:

[1] Un grafo dirigido es aquel en el que sus aristas tienen flechas en los extremos, éstas pueden ser en una o ambas direcciones.

Sean Q el conjunto de todos los estados, Σ un alfabeto de entrada y α un carácter que pertenece a Σ y δ una función o regla de transformación, entonces δ es de la forma $\delta : Q \times \Sigma \to Q$. La tabla 1 describe como es analizada la cadena ab con el diagrama de la figura 1.

δ	a	b
q_1	q_2	$\times$
q_2	q_1	$\times$

Tabla 1: Tabla de transición de estados del diagrama de la figura 1.

Se puede apreciar que al analizar la cadena ab, primero se pasa de q_1 a q_2 por medio de la arista a en el sentido indicado por la flecha, después cuando se analiza b se pasa de q_2 a q_1, por medio de la arista b, la cadena a sido analizada por completo y q_1 es un estado de aceptación, por lo que la cadena es aceptada como parte del lenguaje descrito por $(ab)^*$. Si tratamos de analizar otra cadena como aaa, podemos ver que no hay una transición definida para cuando uno está en el estado q_2 y se analiza una a, por lo que se permanece en este estado. El cual no es un estado de aceptación y la cadena nunca termina de ser analizada, por lo que estamos seguros que aaa no pertenece a el lenguaje descrito por la expresión regular $(ab)^*$.

Supongamos que existe una máquina imaginaria que puede realizar la tarea de analizar cadenas y decir si éstas pertenecen a cierto lenguaje o no. Este tipo de máquinas son conocidas como *máquinas de estados finitos* o *autómata finito*(AF). El cual se define formalmente como una 5-tupla, $M = (Q, \Sigma, F, s, \delta)$ donde:

- Σ, es algún alfabeto de entrada.
- S, es el conjunto finito de todos los estados.
- $F \subseteq S$, es el conjunto de estados finales o de aceptación.
- $s_1 \in S$, es el estado inicial de la máquina(AF)
- δ, es la función de transición de estados $\delta : S \times \alpha \to S$.

Existen varios tipo de autómatas, algúnos de ellos serán brevemente descritos en la sección 2.6.2, indicando que tipo de lenguaje es capaz de reconocer cada uno. Así como hay diferentes tipos de autómata, existen diferentes tipos de lenguajes y cada uno de estos lenguajes pueden ser reconocidos por algún tipo de autómata.

Las diferencias entre los lenguajes radican en la forma en la que se estructuran, ésta estructura es definida por algún tipo de *gramática*, este concepto se analizará en la siguiente sección de una manera breve.

2.6 Clasificación de lenguajes

Una de las formas en la que se clasifican los lenguajes es analizar las reglas que lo definen, es decir su gramática. Los diferentes tipos de gramática serán descritos brevemente en la sección 2.6.1.

2.6.1 Gramáticas

Una gramática es un conjunto de reglas que sigue un lenguaje para la estructura de palabras válidas. Las gramáticas por lo regular producen cadenas o palabras que pertenecen a algún lenguaje, y la producción de estas palabras está determinada por reglas de sustitución bien definidas. A estas reglas también se les llama producciones. Dicho lo anterior, podemos decir que un lenguaje es descrito por una gramática G, donde G es una túpla de cuatro elementos $G = (N, S, \Sigma, P)$, donde:

- N. Un conjunto de símbolos no terminales, que funcionan como variables a través de las reglas de producción. Son substituidas, en cada paso, por una secuencia de símbolos que pueden ser terminales, no terminales o una combinación de ambos.

- $S \in N$. Es el símbolo inicial del cual derivan las demás producciones por medio de las reglas de producción.
- Σ. Un alfabeto sobre el cual se generará el lenguaje. A los símbolos de este alfabeto se les llama terminales, ya que una vez que forman parte de la palabra producida, no son reemplazados por algún otro símbolo.
- P. Es el conjunto de reglas de producción, con el cual se generarán las posibles cadenas que integren el lenguaje.

Podemos pensar en las gramáticas como las reglas que estructuran el lenguaje. Es común que se utilicen letras mayúsculas para denotar a los símbolos no terminales y letras minusculas para denotar a los símbolos terminales. El lenguaje generado por una gramática G comunmente se denota como $L(G)$.

2.6.2 Jerarquía de Chomsky

Existen cuatro tipos importantes de gramáticas, las cuales fueron definidas por Chomsky en [CN65], de acuerdo a la forma de sus producciones. Según Chomsky a cada uno los lenguajes producidos por cada una de éstas gramáticas le corresponde un tipo de autómata que reconozca dicho lenguaje, las correspondecias son las siguientes:

Tipo 0 o *no restringidas.* Este tipo de gramáticas son de la forma $\alpha \rightarrow \beta$ donde $\alpha \in (N \cup \Sigma)^+$ y $\beta \in (N \cup \Sigma)^*$. Estas generan lenguajes *recursivamente enumerables*, que son aceptados por las *máquinas de Turing*, que es un autómata que lee y escribe en una cinta infinita. Hay dos tipos principales de *máquinas de Turing*, las *determinístas* y las *no determinístas*, la diferencia entre ellas es, que la determinísta, solo puede pasar a una configuración siguiente dados un estado actual y una entrada en la cinta, por otro lado, la no determinísta tiene un conjunto finito de configuraciones siguientes a las que puede llegar a partir de una estado actual y una entrada en la cinta. En un movimiento la máquina puede leer, escribir, y desplazarse en hacia la derecha o la izquierda de la cinta. La *máquina de Turing* parará siempre que llegue a un estado final. La secuencia de movimientos que conducen a una configuración de parada recibe el nombre de computación.

Existen algunas variaciones de este tipo de máquina. Aunque ninguna tiene más potencia para realizar una computación que la *máquina de Turing* original, hay casos en los que resulta más conveniente y cómodo utilizar algúna de sus variantes. Cualquiera de sus variantes puede ser emulada por la máquina original.

Tipo 1 o *sensibles al contexto.* Este tipo de gramáticas son de la forma $\alpha \rightarrow \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y la longitud de una cadena generada debe ser mayor o igual a la longitud de la cadena de la cual deriva, esto es, si $\alpha \rightarrow \beta$ entonces $\mid \beta \mid \geqslant \mid \alpha \mid$. Estas gramáticas generan lenguajes, aceptados por los *atómatas linealmente acotados*, que son una variación de una máquina de *Turing no determinista*, solo que se le agrega al alfabeto de la cinta los signos delimitadores '$\langle$' y '$\rangle$' , los cuales acotarán el área de trabajo en la cinta. La producción $S \rightarrow \varepsilon$ es válida solo si la palabra vacía es incluida en el lenguaje.

Hay que recordar que cualquier variación de un *máquina de Turing* puede ser emulada por la máquina de la definición original.

Tipo 2 o *libres de contexto.* Este tipo de gramáticas son de la forma $\alpha \rightarrow \beta$, donde $\alpha \in N$ y $\beta \in (N \cup \Sigma)^*$, el lado izquierdo de su producción (α) consiste en un solo símbolo no terminal y el lado derecho (β) puede ser una combinación de terminales con no-terminales. Éstas producen lenguajes que son aceptados por los *atómatas de pila*, que es una máquina de estados finitos con un medio de almacenamiento infinito que funciona como una pila, este autómata puede apilar y desapilar elementos de la pila dependiendo de el estado actual, el símbolo de la cima de la pila, y el símbolo que se desea introducir.

Un autómata de pila es una tupla de 7 elementos $(Q, \Sigma, \Gamma, s, z, F, \Delta)$ donde:

- Q es un conjunto de los estados a los que cambiará el autómata, después de aplicar la regla de transición.

- Σ es el alfabeto de entrada, son los símbolos que serán empilados.
- Γ es el alfabeto de la pila, la cual debe contener al menos un símbolo al principio.
- $s \in Q$ es el estado inicial del autómata.
- $z \in \Gamma$ es el símbolo inicial en la pila.
- $F \subseteq Q$ es un conjunto de estados finales o de aceptación.
- Δ es la función de transición $\Delta : Q \times \Sigma^* \times \Gamma \to Q \times \Gamma^*$, la cual resulta en un conjunto de pares ordenados de la forma (p, γ) donde $p \in Q$ es el estado siguiente y $\gamma \in \Gamma^*$ es la cadena que será empilada. Un *autómata de pila* acepta una cadena como parte del lenguaje cuando llega a un estado final o de aceptación $q_n \in F$ o termina su operación con la pila vacía.

Tipo 3 o *regulares.* Las producciones de este tipo de gramáticas son de la forma $\alpha \to c$, donde $\alpha \in N$ y c cumple con que, $c = \beta T$ donde $\beta \in \Sigma^*$ y $T \in N$. Sólo tienen un no terminal α en el lado izquierdo de la producción, el lado derecho de la producción debe contener una cadena formada de símbolos terminales y no terminales teniendo como máximo un no terminal que debe estar al extremo derecho de la cadena. Opcionalmente se puede invertir la definición diciendo que se debe contener un no terminal al extremo izquierdo de cada producción, esta variación en la definición invierte el lenguaje generado por la gramática.

A las gramáticas que tiene el no terminal en el extremo derecho de la producción se les llama *lineales por la derecha*, las que lo tienen en el extremo derecho se les llama *lineales por la izquierda.*

El lenguaje producido por este tipo de gramáticas puede ser reconocido por un *autómata finito* como el mencionado en la sección 2.5.

Nota: Un aspecto esencial de la jerarquía de Chomsky es que las gramáticas de nivel $n-1$ contienen a las de nivel n, es decir, las gramáticas *no restringidas* contienen a las gramáticas *sensibles de contexto*, que a su vez contienen a las gramáticas *libres de contexto*, que a su vez contienen a las gramáticas *regulares.*

Dicho lo anterior, los lenguajes generados por gramáticas del nivel $n-1$ contienen a los lenguajes generados por gramáticas de nivel n y, si un autómata acepta el lenguaje generado por una gramática de nivel $n-1$, entonces aceptará el lenguaje generado por la gramática de nivel n.

3 Algebra regular

Ahora que hemos llegado hasta aquí, los conceptos analizados en las secciones anteriores nos serán de mucha utilidad para introducir los aspectos básicos de el *algebra regular*, examinada por John H. Conway en [JC71, cap. 3], la cual parte de la teoría de los experimentos de Moore.

3.1 Teoría de los experimentos de Moore

Comenzaremos esta sección introduciendo la definición formal de una *máquina de estado finitos* o *autómata finito.* De acuerdo con Moore, una *máquina de estados finitos* o *autómata finito* es una 6-tupla $M = (S, I, O, t, o, i)$, donde:

- $S = \{s_1, s_2, \ldots, s_n\}$, es un conjunto finito de estados, para denotar que son los estados la máquina M se utilizará $S(M)$.
- $I = \{i_1, i_2 \ldots, i_m\}$, un conjunto finito de entradas que serán analizadas por alguna máquina M, para denotar que éstas entradas corresponden M se usará la notación $I(M)$.
- $O = \{o_1, o_2 \ldots, o_j\}$, es un conjunto finito de palabras que son resultados o salidas que entregara algúna máquina, para denotar que estas salidas corresponden a la máquina M se usará la notación $S(M)$.
- $t : S \times I \to S$, es una función de transición de estados como la explicada en la sección 2.5.

- $o : S \to O$, es llamada la función de salida. Una máquina en estado s, esataŕa arrojando una salida $o(s)$ hasta que reciva una entrada i, entonces pasará al estado s_i y comenzará a emitir la salida $o(s_i)$.

- i, un estado especial, llamado estado inicial de la máquina.

Para dar una mejor idea de como imaginarse esto, hay que pensar que en algún momento, hay una máquina que se encuentra en el estado s, y emitiendo una salida $o(s)$. Esta máquina permanecerá en este estado, hasta que reciba una entrada w, entoncés cambiará su estado inmediatamente a s_w, y comenzará a emitir una salida $o(s_w)$. Inicialmente la máquina se encuentra en el estado i. Por comodidad se abrevia $t(s_n, w)$ como s_{nw}.

Al igual que como hicímos con los autómatas finitos, es posible, y más comodo describir el comportamiento de una máquina por medio de una tabla, especificando las trancisiones de estado y dando los valores a t, para cada $o(s)$, o con un diagrama de estados. En el último caso tendremos un nodo para cada estado, indicando su nombre y la salida que provoca, y una linea dirigida, marcada con a que conecta el nodo s_n con s_{n+1}, siempre que $s_{na} = s_{n+1}$. El estado inicial esta marcado con una flecha entrante ($\to$). Para el caso particular de una máquina binaria, es decir, con sólo dos salidas diferentes, se puede indicar las salidas en 1 con una flecha saliente($\to$), tal como se muestra en la figura 2.

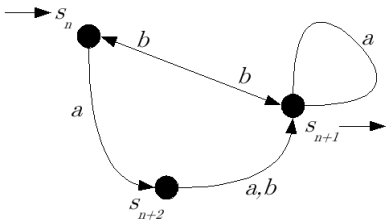

Figura 2: Diagrama de una máquina de tres estados, con dos salidas (0,1).

La tabla 2 corresponde a las transiciones de la máquina presentada en la figura 2, como se puede observar, es muy parecida a la tabla que describe a un AF, con la única diferencia que esta máquina arroja una salida dependiendo de su estado, en este caso sólo puede ser 0 ó 1, dado que es una máquina binaria.

t	a	b	o
s_n	s_{n+2}	s_{n+1}	0
s_{n+1}	s_{n+1}	s_n	0
s_{n+2}	s_{n+1}	s_{n+1}	1

Tabla 2: Tabla de transiciones de estado parar la máquina de la figura 2.

En la teoría de Moore, se considera que en cada experimento, la persona que experimenta con la máquina, no puede saber el estado de la máquina por otro medio que no sea su salida.

Ajustaremos un poco las definiciones de la seccion 2.1, de tal manera que ahora una palabra será una secuencia finita de entradas. La longitud de la palabra, es la logitud de dicha secuencia, y la palabra vacía 1, es la palabra de longitud 0. I^* es el conjunto de todas la palabras de entrada en I. Si s_n es un estado y $w = abc\ldots\ k$ una palabra, entoncés, s_{nw} denota el estado alcanzado desde s_n, al aplicar la secuencia $abc\ldots\ k$ en orden. Si la entrada de una máquina en algún estado s_n es la cadena vacía, la máquina no cambia de estado, $s_{n1} = s_n$. Un estado s_{n+m} es *accesible* desde otro estado s_n, sólo si, exite una palabra w, de tal manera que $s_{nw} = s_{n+m}$, y es *accesible* si éste es accesible desde i, el estado inicial.

También llamaremos a dos estados s_n y s_m *indistinguibles*, si se cumple que $o(s_{nw}) = o(s_{mw})$ para toda palabra w. Esta definición será útil aún si se trata de dos máuqinas diferentes, ya que sólo se necesita que

dichas máquinas tengan el mismo conjunto de entradas $I(M)$ y salidas $O(M)$. Más adelante se verá que, a veces, es posible que una máquina reemplace a otra, sólo si, éstas dos se comportan de la misma manera, bajo las mismas entradas. Esto puede ser bueno si la nueva máquina es menos compleja.

La mayor parte de la notación de esta sección es tomada de [JC71], así que S viene de la palabra en íngles *state*, al igual que las demás letras utilizadas.

3.1.1 Experimentos

En la teoría de los experimentos de Moore, un experimento se define como una función $e : O^* \rightarrow I \cup A$, donde O es el conjunto de todo los resultados que puede entregar la máquina, I es son las entradas y A es un conjunto de *respuestas* donde $I \cap A = \emptyset$.

Para aplicar la función e sobre alguna máquina M, es necesario poner a la máquina a analizar una cadena $w = abc \ldots k$, después de analizarla, la máquina nos proporcionará una palabra, resultado de aplicar la función f que analiza la cadena w, partiendo de un estado s.

$$f(s, w) = o(s), o(s_a), o(s_b) \ldots, o(s_k)$$

A esta palabra resultante se le aplica la función $e(f(s, w))$, si el resultado l esta en I, se concatena w con l y se vuelve a procesar ahora la cadena wl en la máquina con $f(s, wl)$, en caso de que $l \notin I$, indica que $e(f(s, wl))$ es una respuesta, llamada *salida* de e en s.

El objetivo de realizar estos experimentos es saber si podemos distinguir entre dos estados de una máquina analizando unicamente la salida, para esto se experimenta con dicha máquina de tal manera de que se pueda distinguir un estado de otro sólo analizando la longitud de la palabra resultante del experimento.

Se definen diferentes tipos de máquinas dependiendo de las restricciones para el número de estados, entradas y salidas, pero la forma de distinguir estados de diferentes máquinas es similar, se analiza la longitud de la palabra resultante de los experimentos realizados sobre las diferentes máquinas. Incluso se puede determinar con un experimento, si una máquina pertenece o a un conjunto $M = \{m_1, m_2 \ldots, m_s\}$ de máquinas, para esto es necesario que los estados de todas y cada una de las máquinas puedan ser distinguidos.

Como ya mencione la teoría de los experimentos de Moore, se centra en la capacidad de distinguir en entre estados que pueden ser de una misma máquina o de distintas máquinas, ésto nos da la posibilidad de decir que si las salida o resultados de dos máquinas son indistiguibles para toda salida, entonces las dos máquinas realizan el mismo análisis, ésto no implíca que la complejidad de las dos sea la misma.

Si se pueden obtener de dos máquinas diferentes se pueden obtener los mismos resultados y cada uno de los estados de estas máquinas son indistinguibles, entonces se dice que hay isomorfismo entre éllas y tendrán el mismo comportamiento. Sean M y N dos máquinas y α y β los conjuntos estados de M y N respectivamente y para toda cadena o entrada w que se le aplique a la máquina, la función de salida $o(\alpha_w) = o(\beta_w)$, tenemos dos máquinas con el mismo comportamiento y por lo tanto isomórficas.

Ahora podemos decir que es posible que algunos autómatas de los mencionados en la sección 2.6.2, pueden comportarse igual que algún autómata de otro tipo, y ésto concuerda con la jerarquía de Chomsky. Por lo tanto, aunque una máquina de mayor complejidad pueda realizar algún analizis, bien podría utilizarse una máquina más simple sin perdida de eficiencia.

3.2 Teoría de eventos regulares de Kleene y expresiones

En esta sección podremos veremos como es posible representar el comportamiento de un autómata finito por medio de una matriz de transición de estados cuyas entradas son expresiones regulares que definen un lenguaje cada una. A partir de la matriz de transición, podemos conocer que lenguaje aceptará algún autómata dependiendo de su estado inicial y su estado o estados finales, de hecho nosotros podemos indicar cual queremos que sea su estado inicial y su estado o estados finales y así especificar el lenguaje que aceptará dicho autómata. Antes de entrar en detalles, veremos una introducción a la teoría de eventos regulares definida por Stephen Cole Kleene.

Tomemos una máquina M con estado inicial i. En la teoría de eventos regulares de Kleene se define $E(i) = \{w \mid o(i_w) = i\}$, como el conjunto de todas las cadenas que lleven a una máquina M de su estado inicial i, a estados en los cuales la salida de la misma sea $O(M) = i$. De esta manera el comportamiento de dicha máquina esta caracterizado por los conjuntos $E(i)(i \in O(M))$. Se dice que un conjunto E es *representable* si, existe una máquina M y una salida $i \in O(M)$ de tal manera que $E(i) = E$, es decir, E esta formado por todas las cadenas que cumplen con la definición de $E(i)$.

3.2.1 Definiciones

Se ha mencionado que una cadena es una secuencia, posiblemente vacía de símbolos o entradas, ahora definiremos un evento como un conjunto arbitrario de cadenas. No podemos distinguir algún símbolo de la palabra de longitud 1 correspondiente a dicho símbolo, tampoco podemos distinguir alguna palabra y el correspondiente evento de cardinalidad 1, pero podemos distinguir entre el evento vacío que no contiene palabras, y el evento unitario que contiene unicamente la cadena vacía ε. Llamaremos a w una ocurrencia del evento E si, $w \in E$.

El objetivo de Kleene era describir la relación entre un organismo animal y su medio ambiente. El organismo puede detectar algún evento en su medio ambiente, unicamente observando algunas propiedades especiales de los estímulos que recibe. Para propósitos matemáticos, podemos identificar un evento a partir de las secuencias de entrada (estímulos), las cuales corresponden a la ocurrencia de algún evento.

Ahora definiremos las operaciones algebraicas básicas que se pueden realizar con eventos:

- La suma de dos eventos $A + B$, se define igual que la unión de conjuntos $A \cup B$. La suma de un número arbitrario de eventos se define como

$$\sum_{t}^{T} E_t = \bigcup_{t}^{T} E_t$$

- El producto de dos eventos $A \cdot B$, es el conjunto$\{ab, a \in A \text{ y } b \in B\}$, igual que en la concatenación de lenguajes.

Ahora podemos definir la estrella de E, como $E^* = 1 + E^1 + \ldots$, que es la suma de las potencias de E. Las potencias E^n se definen inductivamente como, $E^0 = 1$, $E^{n+1} = E^n \cdot E$. La n-ésima suma parcial de las potencias de E es $E^{<n} = E^0 + E^1 + \ldots + E^{n-1}$, también podemos usar $E^{\leqslant n}$ para denotar $\sum_{i=0}^{n} E^i$ $(i \leqslant n)$, $E^{>n}$ para $\sum_{i>n} E^i$ $(i > n)$, etc.

3.2.2 Operaciones regulares

El conjunto de operaciones $\{+, \cdot, *\}$, son llamadas *operaciones regulares*. Por conveniencia se utiliza el evento 0 como una *operación regular nula*.

Las *operaciones regulares* satisfacen los siguientes axiomas:

1. $A + 0 = A$
2. $A + B = B + A$
3. $(A+B)+C = A+(B+C)$
4. $A \cdot 0 = 0$
5. $0 \cdot A = 0$
6. $A \cdot 1 = A$
7. $1 \cdot A = A$
8. $A(B + C) = AB + AC$
9. $(B + C)A = BA + CA$
10. $(AB)C = A(BC)$
11. $(A + B)^* = (A^*B)^*A^*$
12. $(AB)^* = 1 + A(AB)^*B$
13. $(A^*)^* = A^*$
14. $A^* = A^{n*}A^{<n}(n > 0)$

Los axiomas del 1-10 son fáciles de analizar, pero explicare los últimos 4. Primero, el axioma 11 es $(A + B)^* = (A^*B)^*A^*$, si desarrollamos las potencias de esta expresión, obtenemos $1 + A + B + AA + AB + BA + BB + \ldots$, que es la suma de todos los productos de A's y B's. Del otro lado de la igualdad

tenemos $(A^*B)^*A^*$, la cual podemos expresar como $(A^iB)(A^jB)\ldots(A^mB)A^n$, que es el producto de 0 o más potencias de A's particionadas por la ocurrencia de B's. De manera similar tenemos que

$$(AB)^* = 1 + AB + ABAB + \ldots = 1 + A(AB)^*B,$$

con lo que se demuestra el axioma 12. Para $(A^*)^*$ podemos ver sencillamente que $\cup_{j=0}^{\infty}\cup_{i=0}^{\infty}(A^i)^j = \cup_{n=0}^{\infty}A^n$. El axioma 14 es muy parecido al 13, nos dice que podemos escribir cualquier potencia de A, digamos A^t, como $(A^j)^n.A^k$, donde $0 \leq k < n$. Hay que mencionar que es común abreviar $(A^n)^*$ como A^{n*}, los exponentes serán tratados como si fueran índices, así que, en adelante podríamos utilizar $A^{n*+\leq n}$ en vez de $(A^n)^* \cdot A^{\leq n}$.

Llamaremos a un evento, regular, sólo si puede ser obtenido de los eventos $0, 1$ y las entradas, por medio de la aplicación repetida de las *operaciones regulares* $\{+, \cdot, *\}$, es decir, son un *función regular* de sus entradas. Al hablar acerca de un evento regular estamos hablando al mismo tiempo de un concepto tratado en la sección 2.4, una exresión regular.

Ejemplo: Tomemos a $E = (aa^*ba)^*$, podemos ver que es regular, ya que esta formado por los productos (posiblemente vacíos) de todas las cadenas de la forma aa^nba $(n \geq 0)$, este evento es representable, ya que el evento E está constituido por todas las cadenas w de la forma $(aa^*ba)^*$, que logran que la máquina pare en q_1, un estado en el cual $o(q_{1w}) = 1$, esto es, $E(1) = E$. La figura 3 muestra la representación y la tabla 3 muestra las transiciones de estados de la máquina.

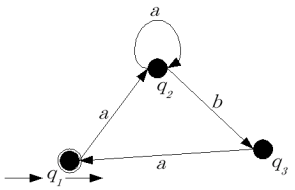

Figura 3: Representación de $E = E(1) = (aa^*ba)^*$.

t	a	b	o
q_1	q_2	$\times$	1
q_2	q_2	q_3	0
q_3	q_1	$\times$	0

Tabla 3: Tabla de transiciones, para la figura 3.

3.2.3 Eventos representables

Cuando un evento puede ser descrito con algún tipo de notación, entonces decimos que dicho evento es representable. El teorema principal de ésta teoría dice:

Teorema 1. *Un evento E es representables si, y sólo si, este es regular.*

La demostración de este teorema implica la demostración de que, la familia de las matricez cuadradas de dimensiones $n \times n$, forman una algebra de Kleene bajo las *operaciones regulares*. Esta demostración es extensa, por lo que se no se verá su totalidad en este documento, por lo que, sí el lector es interesado en examinarla, es posible encontrar dicha demostración en [JC71, cap. 3].

Conway escribió que, en cualquier máquina M se define $E(\alpha, \beta) = \{w \mid \alpha_w = \beta\}$ como el conjunto de todas las palabras que lleven del estado α al estado β. También se define $E_l(\alpha, \beta) = \{w \mid \alpha_w = \beta,$ y w tenga longitud $l\}$. Un evento de la matriz de transición esta definido por $M_{ij} = E_1(\alpha_i, \alpha_j)$ el conjunto de entradas que lleven del estado α_i al estado α_j en una sola transición. Se usará la misma letra

para representar a una máquina M y a la matriz de transición que describe su comportamiento y cuyas entradas son eventos. Para representar un evento regular definido por LM^*N, se debe definir L y N como sigue:

L es un vector fila, que es el vector de entrada, el cual indica el estado inicial de la máquina, y cuyas entradas estan definidas como sigue parar el caso de una máquina binaria:

$$L_i = \begin{cases} 1, & \text{si } \alpha_\text{i} \text{ es el estado inicial} \\ 0, & \text{para cualquier otro estado} \end{cases}$$

El vector de salida esta definido por $N_j = o(\alpha_j)$. Las entradas de éste vector son salidas de M. En el caso en el que M es una máquina binaria(con salidas 0 ó 1), se pueden identificar las salidas 0 y 1 como los eventos 0 y 1 respectivamente.

De lo anterior se puede observar que $E_l = (M^l)_{ij}$, es el evento compuesto de las cadenas de longitud l que llevan a M de α_i a α_j, de modo que $E(\alpha_i, \alpha_j) = (M^*)_{ij}$, es el evento de las cadenas de cualquier longitud que llevan a M de α_i a α_j, donde M^* esta definida como $1 + M + M^2 + \ldots$, es decir, la suma infinita $\sum_{i=0}^{\infty} M^i$ de las potencias de la matriz. El 1 representa a la matriz identidad de dimensiones $n \times n$. Para poder asegurar que un evento definido por LM^*N es regular, debemos demostrar que las entradas de la matriz M^* son eventos regulares, es decir, deben de ser funciones regulares de sus entradas.

Conway demostró en [JC71, cap. 3], que el sistema de las matrices de dimensiones $n \times n$ bajo las operaciones $\{+, \cdot, *\}$ es cerrado, para ello Conway [JC71] definió una Algebra de Kleene Estandard (S-*algebra*, Standard Kleene Algebra) como un conjunto S con tres operaciones $\{\sum, \cdot, *\}$(llamadas S-*operaciones*), definidas sobre S, y elementos particulares 0 y 1, de tal manera que se cumplieran las siguientes propiedades:

S1 $\sum_t^{\emptyset} E_t = 0$ ($\emptyset$ es el conjunto vacío)

S2 $\sum_s^S \sum_t^{T_s} E_t = \sum_t^U E_t \left(U = \bigcup_s^S T_s \right)$

S3 $E \cdot 1 = 1 \cdot E$

S4 $E(FG) = (EF)G$

S5 $\sum_s^S E_s \cdot \sum_t^T F_t = \sum_{(s,t)}^{S \times T} E_s \cdot E_t$

S6 $E^* = \sum_n^{\omega} E_n (\omega = \{0, 1, 2, \ldots\})$

En $S5$, $T \times S$ indica el producto cartesiano de los conjuntos S y T. La suma $\sum_t^T E_t$ esta definida para todo el conjunto de índices $t \in T$, y denota la suma E_t de todos los eventos de longitud t siempre que $t \in T$. $S6$ denota una definición de E^* y es probable encontrarla continuamente. De lo anterior se tiene que $E_0 + E_1 = \sum_t^{(0,1)} E_t$.

Teorema 2. *Si se define $E \leqslant F$ como $E + F = F$, entonces $\leqslant$ es una relación parcial de orden que es válida en cualquier S-algebra.*

Demostración. Se pueden verificar los axiomas de transitividad, de tal manera que, $E \leqslant F \leqslant E$ es cierto, entonces, si sustituimos obtenemos que $E = E + F = F + E = F$, ya que tienen la propiedad de ser operaciones idempotentes, también se cumple que $X = \sum X = \sum (E_t + X) = \sum E_t + X$ siendo X el menor evento que cumple con $X \geqslant E_t$.
□

Teorema 3. *En cualquier S-algebra E^*G es el menor conjunto F que satisface $F = G + EF$.*

Demostración. Cualquier F cumple con $F \geqslant EF$, por lo tanto $F \geqslant E^n F$ para cada $n \geqslant 0$, de donde tenemos que $F \geqslant E^n G$, ya que $F \geqslant G$, y sumando estas dos desigualdades para todo n, se obtiene que $F \geqslant E^* G$. □

Para encontrar la estrella de una matriz M, cuyas entradas son eventos, utilizaremos la definición de la estrella de un evento $E^* = 1 + EE^*$, esto es posible ya que el sistema de las matrices cuadradas es cerrado sobre las operaciones regulares. 1 representa la matriz identidad.

Teorema 4. *Si tenemos una matriz*

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

sobre una S-algebra, y la particionamos de forma que A y D sean matricez cuadradas, entonces:

$$M^* = \begin{pmatrix} (A + D^*C)^* & A^*B(D + CA^*B)^* \\ D^*C(A + BD^*C)^* & (D + CA^*B)^* \end{pmatrix}$$

Demostración. Definamos

$$M^* = \begin{pmatrix} E & F \\ G & H \end{pmatrix},$$

de tal modo que, desarrolando $M^* = 1 + MM^*$, se obtiene:

$$M^* = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix},$$

que puede ser representado por el siguiente sistema de ecuaciones:

1 $E = 1 + AE + BG$

2 $F = AF + BH$

3 $G = CE + DG$

4 $H = 1 + CF + DH$

Usando el teorema 3, se deduce el siguiente sistema:

(a) $E \geqslant (A + BD^*C)^*$

(b) $F \geqslant A^*B(D + CA^*B)^*$

(c) $G \geqslant D^*C(A + BD^*C)^*$

(d) $H \geqslant (D + CA^*B)^*$

Los valores de este último sistema satisfacen las ecuaciones del primer sistema, por lo que definen una matriz N, de tal manera que $N = 1 + NM$, cuando $N = M^*$. □

Teorema 5. *Las entradas en cualquier M^*, pueden ser expresadas como funciones regulares de las entradas de M.*

Demostración. Para matrices de 1×1, esto es muy sencillo. La manera de hacerlo para matrices de 2×2, es igual a la forma en que se calcula la estrella de una matriz a la en el teorema 4, esto es, utilizando la definición $M^* = 1 + MM^*$. Para una matriz de dimensiones $n \times n$ se puede seguir el siguiente procedimiento:

1. Particionaremos la matriz M en cuatro submatricez, de tal manera que D sea una matriz de dimensiones $(n-1) \times (n-1)$.
2. Ahora se le aplica el paso 1 a la submatriz D y así recursivamente hasta obtener una matriz de dimensiones 2×2. El resultado de particionar a M sera alguna matriz D_i donde i indica el nivel de profundidad de la matriz, es decir, el número de veces que se aplicó el paso 1.

$$M = \left(\begin{array}{c|c} a_{11} & \cdots \\ \hline \vdots & \left(\begin{array}{c|c} a_{21} & \cdots \\ \hline \vdots & \left(\begin{array}{cc} a_{33} & a_{34} \\ a_{43} & a_{44} \end{array}\right)_{2\times 2}^{D_2} \end{array}\right)_{3\times 3}^{D_1} \end{array}\right)_{4\times 4}$$

3. Una vez que se tenga una matriz de 2×2 se calcula su estrella de la misma manera que se calculo ateriormente, esto es $M^* = 1 + MM^*$.

4. Por último, una vez obtenida D_i^*, se puede calcular D_{i-1}^* de la misma forma, y así sucecivamente hasta calcular M^*

Es posible observar que las entradas resultantes de M^* son funciones regulares de las entradas de M, con lo que se demuestra que M^* es regular, y que cada una de sus entradas M_{ij} es una expresión regular que define un lenguaje aceptado por un autómata cuyo estado inicial es aquel definido por L_i y su estado final definido por N_j autómata. □

Hasta ahora se ha demostrado la mitad del teorema 1, y sabemos que cualquier evento representable cumple con que $E(1)$, para alguna máquina binaria, este evento es de la forma LM^*N y además define una expresión regular. Para la segunda parte de la demostración se analiza el hecho de que un evento E, unicamente puede ser representado por una máquina si $E(1) = LM^*N$, donde:

- L es un vector fila *constante* con un sólo 1.
- M es una matriz *lineal*, con cada entrada ocurriendo sólo una vez en cada fila.
- N es un vector columna *constante*.

Un evento es *constante*, si es 0 ó 1, *lineal* si es una suma de entradas, esta definición aplica también a matrices si, y sólo si, aplica a cada una de sus entradas.

Conway se libró de las dos restricciones anteriores al definir un tipo de máquina más general, llamado mecanismo lineal, el cual es un conjunto de objectos α_i llamados nodos, junto con tres funciones:

- Una *función inicial*, la cual asigna para cada α_i un evento L_i.
- Una *función de transformación* que asigna para cada par $(\alpha_i\ ,\ \alpha_j)$ un evento lineal M_{ij}.
- Una *función de salida*, que asigna para cada α_i un evento constante N_i.

Este mecanismo lineal es un tipo especial de máquina, en la cual, en algún momento, un subconjunto de nodos definen el estado en el que se encuentra la máquina. Inicialmente sólo aquellos nodos α_i en los que $L_i = 1$. Un nodo α_j se activa inmediatamente después de que la máquina recibe una entrada a si, y sólo si, había un nodo α_i activo justo antes, y además $a \in M_{ij}$. Al igual que antes, el mecanismo sólo dará como salida 1 cuando exista algún nodo α_j para el cual $N_j = 1$.

El evento representado por el mecanismo lineal es LM^*N, el cual contiene todas las palabras que llevan a la máquina de su estado inicial, a estados en los que $o(\alpha) = 1$.

De manera formal tenemos que los siguiente:

Teorema 6. *E es representado por un mecanismo lineal si, y sólo si, E puede ser representado por una máquina.*

Demostración. Podemos ver cualquier máquina como un mecanismo lineal definido por (L, M, N), tomando los estados de la máquina como los nodos del mecanismo, de esta manera no se afecta el evento representado por la máquina, pero para cada mecanismo lineal (L, M, N), podemos definir una máquina cuyos estados son un subconjunto de nodos α_i, con $\alpha_a = \{\alpha_j | a \in M_{ij}$ para algún i para el cual $\alpha_i \in \alpha\}$, una función de salida $o(\alpha) = 1$, sólo si existen $\alpha_j \in \alpha$ para los cuales $L_j = 1$, y con estado inicial definido por conjunto de nodos α_i en los que $L_i = 1$. La máquina tendrá 2^n estados si (L, M, N) tiene n nodos, y representará el mismo evento que (L, M, N). □

Teorema 7. *Un evento E es representable si, y sólo si, puede ser expresado en la forma $E = L(C + D)^*N$, donde L es un vector fila, N es un vector columna, ambos constantes, C es una matriz cuadrada constante, y D es una matriz cuadrada lineal.*

Demostración. Sabemos que $(C + D)^* = (C^*D)^*C^*$, entonces $L(C + D)^*N = L(C^*D)^*C^*N$, donde $(L, (C^*D)^*, C^*N)$ define un mecanismo lineal, dado que $(C^*D)^*$ es *lineal*, y C^*N es *constante*. □

Teorema 8. *Cualquier evento regular es representable en la forma descrita por el teorema 7.*

Demostración. Se pueden inducir los siguientes resultados de operaciones con eventos regulares, a partir de la demostración del teorema 4, y recordando la forma descrita por el teorema anterior, ésta es, LM^*N.

1. $0 = 0(0)^*0$
2. $1 = 1(1)^*1$
3. $a = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix}^* \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
4. $LM^*N + PQ^*R = \begin{pmatrix} L & P \end{pmatrix} \begin{pmatrix} M & 0 \\ 0 & Q \end{pmatrix}^* \begin{pmatrix} N \\ R \end{pmatrix}$
5. $LM^*N \cdot PQ^*R = \begin{pmatrix} L & 0 \end{pmatrix} \begin{pmatrix} M & NP \\ 0 & Q \end{pmatrix}^* \begin{pmatrix} 0 \\ R \end{pmatrix}$
6. $(LM^*N)^* = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} M & N \\ L & 0 \end{pmatrix}^* \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Lo anterior muestra que 0 , 1 y las entradas (símbolos de entrada) pueden ser expresadas en la forma del teorema anterior y que si E y F son eventos que cumplen con ésta característica, entonces $E + F$ y $E \cdot F$ también lo harán. □

Como un ejemplo tomemos el evento regular $E = (a^*b^*)^*$, para el cual a^* y b^* están son representados por los diagramas de la figura 4.

(a) (b)

Figura 4: (a) a^*, (b) b^*.

$$a^* = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix}^* \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad b^* = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} b & 0 \\ 0 & 0 \end{pmatrix}^* \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Figura 5: Representación matricial de los eventos a^* y b^* en la forma LM^*N descrita en el teorema 7.

De manera que usando la propiedad 5, obtenemos

$$a^*b^* = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \end{array}\right) \left[\begin{array}{cc|cc} a & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & b & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]^* \left[\begin{array}{c} 0 \\ 0 \\ \hline 1 \\ 0 \end{array}\right],$$

y usando la propiedad 6 tenemos:

$$E = \left(\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \end{array}\right) \left[\begin{array}{cccc|c} a & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \end{array}\right]^* \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ \hline 1 \end{array}\right],$$

Así podemos observar que E es representado por el mecanismo de la figura 6(a), a partir de la cual podemos construir la máquina de la figura 6(b), omitiendo los estados inaccesibles. Como se puede observar el estado $\{\alpha, \beta, \gamma\}$ se repite en los tres nodos, proporciona la misma salida (1) en todos los casos, y cada nodo es llevado por medio de a y b al mismo estado, por lo que estos estados son indistinguibles y la máquina puede ser reducida a la de la figura 7.

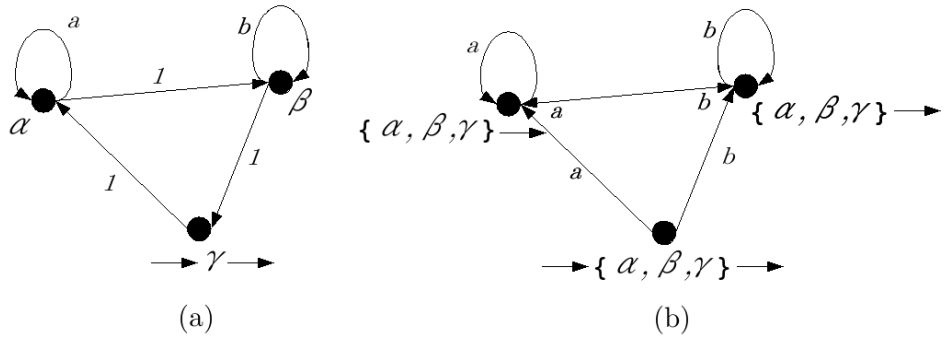

Figura 6: (a) Mecanismo, (b) Máquina; ambos representando $E = (a^*b^*)^*$.

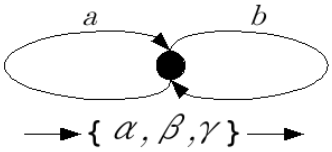

Figura 7: Máquina reducida correspondiente a la figura 6(b).

Uno puede darse cuenta de que el mecanismo de la figura 6(a), no es un *mecanismo lineal*, sino un *mecanismo-(constante+lineal)*, el cual está definido por una tupla de tres elementos $(L, C+D, N)$, en la forma descrita por el teorema 7. En éste tipo de mecanismo, la activación de un nodo α_i causa instantaneamente la activación del nodo α_j para el cual $C_{ij} = 1$. Éste *mecanismo-(constante+lineal)* es equivalente a el mecanismo definido por (LC^*, DC^*, N), el cual por medio del teorema 6, es equivalente a una máquina.

Hasta ahora hemos podido analizar como es que éste mecanismo lineal se asemeja mucho a la máquina de Turing mencionada en las primeras secciones, ya que nos permite emular el comportamientode cualquier máquina, y en el caso de un mecanismo lineal nos deja saber como se comportan dos o más de ellas si éstas son combinadas, probablemente éste principio, de hacer interactuar dos máquinas o mecanismos conociendo unicamente sus salidas se pueda aplicar en otros contextos con resultados interesantes.

Por último, para terminar con la demostración del teorema 1 se necesita probar el siguiente teorema.

Teorema 9. *Existe un procedimiento efectivo para decidir si dos expresiones regulares R y S representan el mismo evento.*

Demostración. Basta con construir dos máquinas M y N que representen a R y S, entonces $R = S$, sólo si las versiones mínimas de M y N son isomórficas, es decir, tienen el mismo comportamiento. □

Nota: La versión mínima de una máquina M es aquella máquina donde cada estado de M es accesible, y cada par de estados distintos son distinguibles.

4 Conclusiones

Conway advierte en su libro, que ésta demostración del teorema de Kleene, proporciona un procedimiento bastante bueno para pasar de máquinas a expresiones regulares, pero muy ineficiente para el caso inverso. Esto es comprobable dado que al aumentar la longitud de la expresión regular que describe su comportamiento, el tamaño de la máquina crece rápidamente, lo que suele resultar catastrófico si no se tiene el cuidado durante éste proceso.

La concepción de una máquina que pueda simular el comportamiento de cualquier otra máquina como el mecanismo lineal definido por Conway [JC71, cap. 3] nos recuerda inmediatamente a la máquina de Turing mencionada al principio de este documento, la cual tiene la capacidad de simular el comportamiento de cualquier tipo de autómata e incluso de simular el comportamiento de todas las variaciones de ella misma.

Habría que ver si esta descripción de mecanismos capaces de simular o embeber el comportamiento de otros mecanismos es aplicable en otros contextos como por ejemplo en la biología, donde por lo regular se sabe el resultado de un experimento, pero raramente se es conciente de el proceso interno que se lleva a cabo para obetner dicha salida.

Agradecimientos

Quiero agradecer al Profesor Harold V. McIntosh y a la Universidad Autónoma de Puebla por su apoyo para participar en el XVI verano de investigación.

Referencias

[CN65] Chomsky, N., *Aspects of the Theory of Syntax.*, MIT Press, Cambridge, MA. (1965)

[DG97] Dafydd Gibbon, *Introduction to Computational Phonology*, AMU English Department, Poznan, `http://www.spectrum.uni-bielefeld.de/Classes/Winter97/IntroCompPhon/compphon/compphon.html` (1-3 December 1997), Version of November 1997.

[DK95] Dean Kelley, *Teoría de atómatas y lenguajes formales*, Traducción: Ma. Luisa Díez Platas, Revición técnica: Luis Joyanes Aguilar, PRENTINCE HALL International(UK)Ltd., Campus 400, Maylands Avenue, Hemel Hempstead, Herdforeshire, HP2 7EZ, Simon & Schuster International Group, **ISBN: 0-13-518705-2**, Depósito legal: M35539-1995, Traducido de : *Automata and formal languages: An introduction*, 302 páginas.

[DK90] Dexter Kozen, *On Kleene Algebras and Closed Semirings*, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501, USA `http://citeseer.ist.psu.edu/cache/papers/cs/1362/http:zSzzSzwww.cs.cornell.eduzSzkozenzSzpaperszSzkacs.pdf/kozen90kleene.pdf` (31 de Mayo, 1990)

[DKL2] Dexter Kozen,*Introduction to Kleene Algebra, (CS786) Lecture 2: Axioms of Kleene Algebra.* `http://www.cs.cornell.edu/Courses/cs786/2004sp/Lectures/l02-axioms.pdf`, (28 de Enero, 2004)

[DKL7] Dexter Kozen, *Introduction to Kleene Algebra, (CS786) Lecture 7: Equational Theory of Kleene Algebra*, `http://www.cs.cornell.edu/Courses/cs786/2004sp/Lectures/l07-complete.pdf`, (16 de Febrero, 2004)

[DKXX] Dexter Kozen, *A completeness Theorem for Kleene Algebras and the Algebra of Regular Events*, Departmente of Computer Science, Cornell University, Ithaca, New York `http://citeseer.ist.psu.edu/cache/papers/cs/3234/http:zSzzSzwww.cs.cornell.eduzSzkozenzSzpaperszSzka.pdf/kozen94completeness.pdf`

[JC71] J. H. Conway, *Regular Algebra and Finite State Machines*, Lecture in Mathematics, University of Cambridge. Chapman and Hall Mathematics Series, *Edited by* Ronald Brown, *University College of North Wales, Bangor*, J. De Wet, *Balliol College, Oxford*, First Published 1971.

[JP02] James Power, *Notes on Formal Language Theory and Parsing*, Department of Computer Science, NATIONAL UNIVERSITY OF IRELAND, Maynooth, Co. Kildare, Ireland, `http://www.cs.may.ie/ jpower/Courses/parsing/` (29 November 2002)

[KD02] Karl Dahlke, *Languages, An Introduction*, from Math Reference Project `http://www.mathreference.com/lan,intro.html` (2002-2006)

[LF02] Mikel L. Forcada, *Neural Networks: Automata and Formal Models of Computation*, Universitat d'Alacant, Dept. Llenguatges i Sistemes Informàtics, E-03071 Alacant, (Spain), `http://www.dlsi.ua.es/ mlf/nnafmc/pbook/` (21 de Enero,2002)

[MM67] Marvin Minsky, *Computation: Finite and infinite machines*, Prentince-Hall, Inc., Engelwood Cliffs, N.J.(1967)

[MW04] Insall, Matt and Weisstein, Eric W., *Lattice*, From MathWorld–A Wolfram Web Resource `http://mathworld.wolfram.com/Lattice.html`(19 de Diciembre,2004)

[MW02] Weisstein, Eric W., *Absorption Law*, From MathWorld–A Wolfram Web Resource `http://mathworld.wolfram.com/AbsorptionLaw.html` (9 de Febrero, 2002