

TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE  
IZTAPALAPA

INTEGRANTES:

CUANENEMI CUANALO MARIO ALBERTO	181080030
FERMIN CRUZ ERIK	181080007
GUTIERREZ ARELLANO RAFAEL	181080022
PEREZ ARMAS FAUSTO ISAAC	181080037

ISC-6AM

LENGUAJES Y AUTOMATAS I

M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

SEP 2020 / FEB 2021

ACTIVIDAD SEMANA 11

Cuanenemi Cuanalo Mario Alberto

# EXPRESIONES REGULARES

## 1.-DEFINICION

## 2.-EQUIVALENCIA ENTRE EXPRESIÓN REGULAR Y AFD

Se entiende por Expresión Regular a la forma compacta de representar los lenguajes. Por lo cual a partir de la expresión regular podemos pasar directamente a su automáta y viceversa.

### DEFINICION:

Si  $A$  es un alfabeto, una expresión regular sobre este alfabeto se define de la siguiente forma:

- $\emptyset$  es una expresión regular que denota el lenguaje vacío ( el que no tiene ninguna cadena) automáta sin ningn estado.
- $\epsilon$  es una expresión regular que denota el lenguaje  $\{\epsilon\}$ .
- si  $a \in A$ ,  $a$  es una expresión regular que denota el lenguaje  $\{a\}$ .
- Si  $r$  y  $s$  son expresiones regularesdenotando los lenguajes  $R$  y  $S$  entonces  $(r+s)$  es una expresión regular que denota el lenguaje  $R \cup S$ .
- Si  $(rs)$  es una expresión regular que denota el lenguaje  $RS$ .
- $r^*$  es una expresión regular que denota el lenguaje  $R^*$ .

A partir de esta definición se puede determinar las expresiones regulares sobre un determinado alfabeto. El tipo de lenguaje al que está asociadas las expresiones regulares son los lenguajes regulares.

...as expresiones regulares se pueden eliminar los paréntesis siempre que no haya dudas, la precedencia de las operaciones es: Clausura  $*$ , Concatenación, Unión.

EJEMPLOS:

- Obtener la expresión regular que representa el lenguaje  $L=\{01^i / i=0\}$   
 $\{01^i / i=0\}=\{0\}\{1^i / i=0\}=\{0\}\{1\}^* = 0 1^*$  la expresión regular correspondiente.
- Obtener la expresión regular que represente el lenguaje cuyas cadenas está formadas por ceros y unos en cualquier posición,  $L=\{u / u \in \{0,1\}^*\}$

Sea  $u=01001$  una cadena con una combinación de ceros y unos cualquiera entonces su correspondiente expresión regular será  $\{0\}\{1\}$ . Como lo que nos piden es ceros y unos en cualquier posición, aplicamos la operación de clausura  $*$  obteniendo:  $(\{0\} \cup \{1\})^*$  por tanto la expresión regular que resulta es la siguiente  $(0+1)^*$

- Dada la expresión regular  $(a+\varepsilon)b^*$  obtener el lenguaje que representa:  
Sea  $(a+\varepsilon)$  el lenguaje 1 notado  $L_1$ . Sea  $b^*$  el lenguaje 2 notado  $L_2$ .  $L_1=\{a\} \cup \{\varepsilon\}$  y  $L_2=\{b\}^*=\{b^i / i=0\}$ , por tanto,  $L=L_1 \cup L_2$  se especifica de la siguiente forma:  $L=\{uv / u \in L_1, v \in L_2\}=\{ab^i / i=0\}$
- Obtener el lenguaje asociado a la expresión regular  $0^*1^*$

Sea  $L_1=0^*$  y  $L_2=1^*$  sus correspondientes expresiones regulares son  $L_1=\{0\}^*=\{0^i / i=0\}$   $L_2=\{1\}^*=\{1^j / j=0\}$  Por tanto,  $L=L_1 L_2=\{0^i 1^j / i, j=0\}$  es el lenguaje regular asociado a la expresión dada.

- Obtener el lenguaje asociado a la expresión regular  $(1+10)^*$

Representa el lenguaje que empieza por uno y no contiene dos ceros consecutivos.

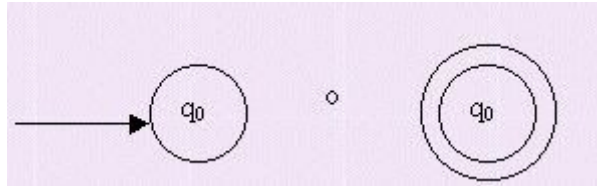
## EQUIVALENCIA ENTRE EXPRESION REGULAR Y AFD.

Teorema: dada una expresión regular existe un automáta finito que acepta el lenguaje asociado a esta expresión regular. Por

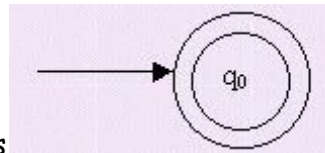
tanto existe una equivalencia entre la expresión regular y el automáta finito determinístico.

Para ello tenemos como base los AFD asociados a las expresiones regulares siguientes:

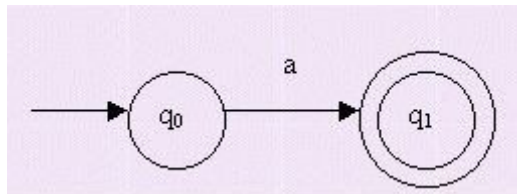
f su automáta es el que no tiene ningún estado, es decir,



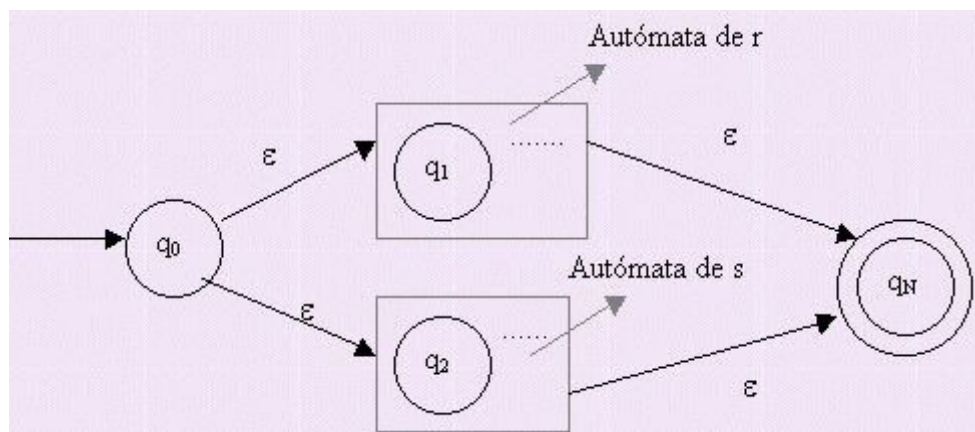
e su automáta asociado es



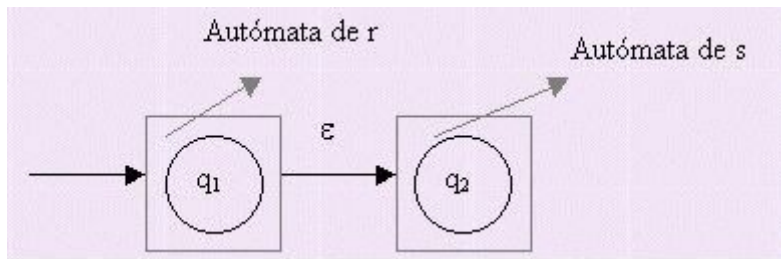
Si  $a \in A$  es una expresión regular su automáta asociado es



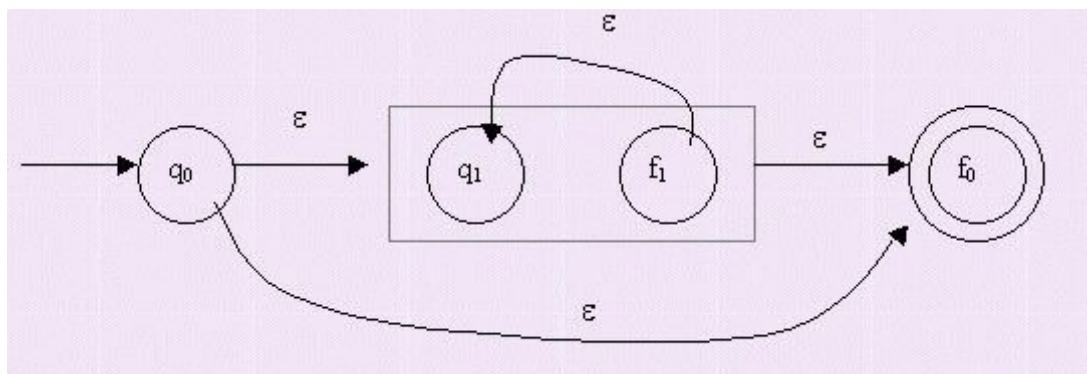
El automáta asociado a la expresión  $(r+s)$  es



... Automáta asociado a la expresión  $(rs)$  es



El automáta asociado a la expresión  $r^*$  es



Conociendo estos automáatas podemos pasar a ver algunos ejemplos para construir AFD que acepten un lenguaje correspondiente a la expresión regular dada.

EJEMPLOS:

Construir el AFD que acepte el mismo lenguaje que el asociado a la expresión regular  $r=01^*+1$ .

Obtenido el AFND CON TN podemos usar algoritmos que transforma:

## ...JD CON TN a AFND

### AFND a AFD

### AFD a AFD MINIMAL

Expresiones y sus lenguajes Estrictamente hablando, una expresión regular  $E$  es sólo una expresión, no un lenguaje. Deberíamos emplear  $L(E)$  cuando deseemos hacer referencia al lenguaje que  $E$  representa. Sin embargo, es habitual emplear " $E$ " cuando realmente lo que se quiere decir es " $L(E)$ ". Utilizaremos este convenio siempre y cuando esté claro que estamos hablando de un lenguaje y no de una expresión regular.

confusiones con la expresión  $0^*$ , cuyo lenguaje son todas las cadenas que constan de un 0 y un número cualquiera de 1s. La razón de esta interpretación se explica en la Sección 3.1.3, pero podemos adelantar que el operador asterisco precede al punto y que por tanto el argumento del asterisco se selecciona antes de realizar cualquier concatenación. Sin embargo,  $L(0^*)$  no es exactamente el lenguaje que deseamos. Sólo incluye aquellas cadenas formadas por 0s y 1s alternos que comienzan por 0 y terminan por 1. También necesitamos considerar la posibilidad de que exista un 1 al principio y/o un 0 al final de las cadenas. Un método sería construir tres expresiones regulares más que manejaran estas tres otras posibilidades. Es decir,  $(10)^*$  representa las cadenas alternas que comienzan por 1 y terminan por 0, mientras que  $0(10)^*$  se puede emplear para las cadenas que comienzan y terminan por 0 y  $1(01)^*$  para las cadenas que comienzan y terminan por 1. La expresión regular completa es  $(01)^* + (10)^* + 0(10)^* + 1(01)^*$ . Observe que utilizamos el operador  $+$  para obtener la unión de los cuatro lenguajes que nos proporcionan todas las cadenas con ceros y unos alternos. Sin embargo, existe otra forma de obtener una expresión regular algo más sucinta.

Partimos de nuevo de la expresión  $(01)^*$ . Podemos añadir un 1 opcional al principio si concatenamos por la izquierda la expresión  $\epsilon + 1$ . Del mismo modo, añadimos un 0 opcional al final con la expresión  $\epsilon + 0$ . Por ejemplo, empleando la definición del operador  $+$ :  $L(\epsilon + 1) = L(\epsilon) \cup L(1) = \{\epsilon\} \cup \{1\} = \{\epsilon, 1\}$ . Si concatenamos este lenguaje con cualquier otro lenguaje  $L$ , la opción  $\epsilon$  nos proporciona todas las cadenas de  $L$ , mientras que la opción 1 nos proporciona  $1w$  para todas las cadenas  $w$  de  $L$ . Por tanto, otra expresión para el conjunto de cadenas formadas por ceros y unos alternos es:  $(\epsilon + 1)(01)^*(\epsilon + 0)$ .



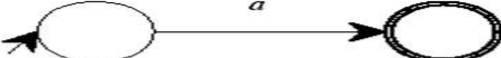
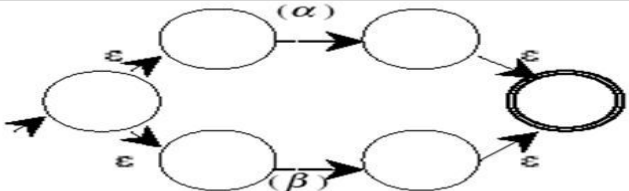
Observe que es necesario encerrar entre paréntesis cada una de las expresiones añadidas, con el fin de garantizar que los operadores se agrupan correctamente.

Fermin Cruz Erik

## EXPRESIÓN REGULAR

Los lenguajes aceptados por un AF son fácilmente descritos por una expresión llamada Expresión regular. Esto quiere decir que:

Sea E un conjunto finito de símbolos y sean L, L1 y L2 conjunto de cadenas de E, la concatenación de L1 y L2, denotada por L1L2, es el conjunto {xy| donde x esta en L1 e Y esta e L2}.

Expresión Regular	Autómata finito
$\emptyset$	
$\epsilon$	
$a \in \Sigma$	
$\alpha   \beta$	

### EJEMPLOS:

Sea  $E = \{0,1\}$  y sea  $R = 0^*1+0$  y queremos construir un autómata cuyo lenguaje sea exactamente el definido por la expresión regular r. El último operador que interviene es la suma.

Gutierrez Arellano Rafael

En **cómputo teórico** y teoría de **lenguajes formales** una expresión regular, o expresión racional, también son conocidas como regex o regexp, por su contracción de las palabras inglesas regular expression, es una secuencia de **caracteres** que conforma un patrón de búsqueda. Se utilizan principalmente para la **búsqueda de patrones** de cadenas de caracteres u operaciones de sustituciones.

Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una **cadena de texto**. Las expresiones regulares proporcionan una manera muy flexible de buscar o reconocer cadenas de texto. Por ejemplo, el grupo formado por las cadenas *Handel*, *Händel* y *Haendel* se describe con el patrón "H(a|ä|ae)ndel".



Carácter	Significado
^	Principio de la cadena
\$	Final de la cadena
.	Cualquier carácter excepto salto de línea
*	Operador de repetición 0 o más veces
+	Operador de repetición 1 o más veces
?	Operador alternativo: una vez o ninguna
	Alternativa
( )	Agrupar expresiones
[ ]	Conjunto de caracteres
{ }	Modificador de repetición
\	Permite presentar un metacarácter como un carácter ordinario

Las expresiones regulares son las grandes olvidadas, no se utilizan mucho, pero cuando te toca utilizarlas lamentas no conocerlas más. Casi toda la gente que conozco dice conocerlas «más o menos», pero cuando le preguntas por un problema concreto no sabe resolverlo.

El problema es que no son intuitivas a primera vista, por lo que la solución a la que llegan muchos programadores con experiencia que no quieren aprenderse la sintaxis a fondo es tener un conjunto de soluciones que le han servido en el pasado y partir de alguna que se parezca.

Lo que voy a intentar con este tutorial es por una parte, dar una base a la gente que esté empezando, y por otra, servir de referencia para la gente que tiene una idea general pero necesita algo específico.

```

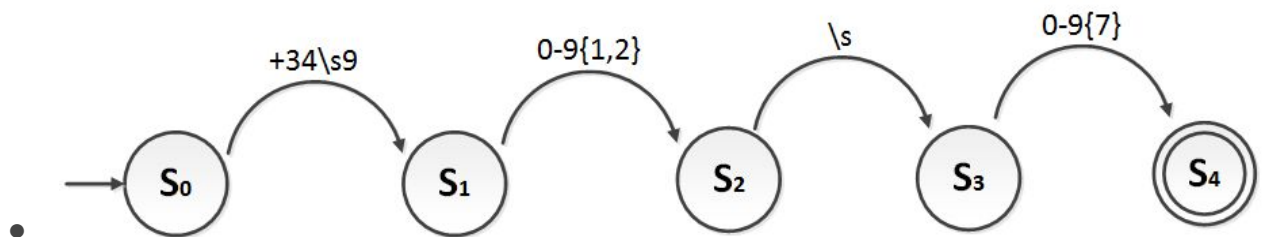
: / \ [aA-zZ] .+? \ graciasform \ .html \ ?value= ([1-9] ( .+ ) ? )
TEST STRING
51
/google-analytics/graciasform.html?value=100
/organic-search/graciasform.html?value=10
/organic-search/graciasform.html?value=70
/organic-search/graciasform.html?value=66
/afiliates-course/graciasform.html?value=50

```



..... ve lista de los más utilizados:

- $\wedge$  Indica el principio de una cadena
- $\$$  Indica el final de una cadena
- $()$  Un agrupamiento de parte de una expresión
- $[]$  Un conjunto de caracteres de la expresión
- $\{ \}$  Indica un número o intervalo de longitud de la expresión
- $.$  Cualquier carácter salvo el salto de línea
- $?$  0-1 ocurrencias de la expresión
- $+$  1-n ocurrencias de la expresión
- $*$  0-n ocurrencias de la expresión
- $\backslash$  Para escribir un carácter especial como los anteriores y que sea tratado como un literal
- $|$  Para indicar una disyunción lógica (para elegir entre dos valores:  $a|b$  se tiene que cumplir al menos uno de los dos)
- 



Aprendiendo con ejemplos

La forma más rápida para aprender a hacer expresiones regulares es mediante ejemplos, en el link de abajo tenéis una página para ir probando combinaciones y comprobar en tiempo real el resultado.

**Perez Armas Fausto Isaac**

Una expresión regular es un generador de lenguajes sobre un alfabeto con ciertas características (restricciones).

ción de las expresiones en programación, definen a las ER como:

Las expresiones regulares se usan para analizar el contenido de cadenas de caracteres por medio de patrones.

Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.

Si  $\Sigma$  es un alfabeto,  $\Sigma^*$  denota el conjunto de todas las cadenas sobre el alfabeto y se conoce como cerradura de  $\Sigma$  o lenguaje universal sobre  $\Sigma$

$\Sigma^*$  es infinito para cualquier  $\Sigma$

Ejemplos de expresiones regulares:

1.- Cadenas sobre  $\{a,b\}$  que contengan bb

$$L = \{a,b\}^* \{bb\} \{a,b\}^*$$

2.- cadenas que inician con aa o terminan con bb sobre  $\{a,b\}$

$$L = \{aa\} \{a,b\}^* \cup \{a,b\}^* \{bb\}$$

3.- Si  $L_1 = \{bb\}$  y  $L_2 = \{e, bb, bbbb\}$  son lenguajes sobre  $\{a,b\}$

$L_1^*$  y  $L_2^*$  representan cadenas con numero par de b's

4.-  $\{aa, ab, ba, bb\}$  son cadenas de longitud par sobre  $\{a,b\}$

$$\{a,b\} \{aa, ab, ba, bb\} y$$

$\{aa, ab, ba, bb\} \{a,b\}$  son cadenas de longitud non

5.- Cadenas que inician y terminan con a y contienen al menos una b

$$\{a\} \{a,b\}^* b \{a,b\}^* \{a\}$$

6.-  $(a \cup b)^* aa (a \cup b)^*$  cadenas con aa

$$(a \cup b)^* bb (a \cup b)^* \quad \text{cadenas con bb}$$

$$(a \cup b)^* aa (a \cup b)^* \cup (a \cup b)^* bb (a \cup b)^*$$

cadenas con aa o bb

7.- Contienen exactamente dos b's sobre  $\{a,b\}$

$$a^* ba^* ba^*$$

as con al menos dos b's

$$(a \cup b)^* b (a \cup b)^* b (a \cup b)^*$$

9.- Un numero par de b's

$$a^* (a^* b a^* b a^*)^*$$

10.- Sobre {a,b} que no contengan aa

$$b^* (ab)^* \cup b^* (ab)^* a$$

$$(b \cup ab)^* \cup (b \cup ab)^* a$$

11.- Cadenas que contienen bc sobre {a,b,c}

$$(a \cup b \cup c)^* bc (a \cup b \cup c)^*$$

12.- Cadenas sobre {a,b,c} que no contienen bc

$$c^* (b \cup ac^*)^*$$

Principales equivalencias

$$r \cup s = s \cup r$$

$$(r \cup s) \cup t = r \cup (s \cup t)$$

$$r \cup \emptyset = \emptyset \cup r = r$$

$$r \cup r = r$$

$$r\epsilon = \epsilon r = r$$

$$r\emptyset = \emptyset r = \emptyset$$

$$(rs)t = r(st)$$

$$r(st) = rs \cup rt$$

$$(r \cup s)t = rt \cup st$$

$$r^* = r^* r^* = (r^*)^* = (\epsilon \cup r)^*$$

$$\emptyset^* = \epsilon^* = \epsilon$$

$$r^* = \epsilon \cup r r^*$$

$$(r \cup s)^* = (r^* \cup s^*)^* = (r^* s^*)^* = r^* s^* r^* = r^* (s r^*)^*$$

$$r(sr)^* = (rs)^* r$$

$$(r^*s)^* = \varepsilon \cup (r \cup s)^*s$$

$$(rs^*)^* = \varepsilon \cup r(r \cup s)$$

$$1.- (ba^+)(a^*b^* \cup a^*) = (ba)^* b a^+ (b^* \cup \varepsilon)$$

$$=(ba)^* b a a^* (b^* \cup \varepsilon)$$

$$=(ba)^* b a a^* (b^* \cup \varepsilon)$$

$$=(ba)^+ a^* (b^* \cup \varepsilon)$$

$$=(ba)^+ (a^* b^* \cup a^*)$$

$$2.- (a \cup b)^* = (b^* (a \cup \varepsilon) b^*)^*$$

$$= (b^* a \cup b^*) b^*)^*$$

$$= (b^* a b^* \cup b^* b^*)^*$$

$$= (b^* a b^* \cup b^*)^* (r \cup \varepsilon)^* = (r^*)^* = r^*$$

$$= b^* (a b^* \cup \varepsilon)^*$$

$$= b^* ((a b^*)^*)^*$$

$$= b^* (a b^*)^*$$

$$r^*(sr^*)^* = (r \cup s)^*$$

$$= (b \cup a)^*$$

$$= (a \cup b)^*$$