TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INTEGRANTE:

PEREZ ARMAS FAUSTO ISAAC

181080037

ISC-6AM

LENGUAJES Y AUTOMATAS I

M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

SEP 2020 / FEB 2021

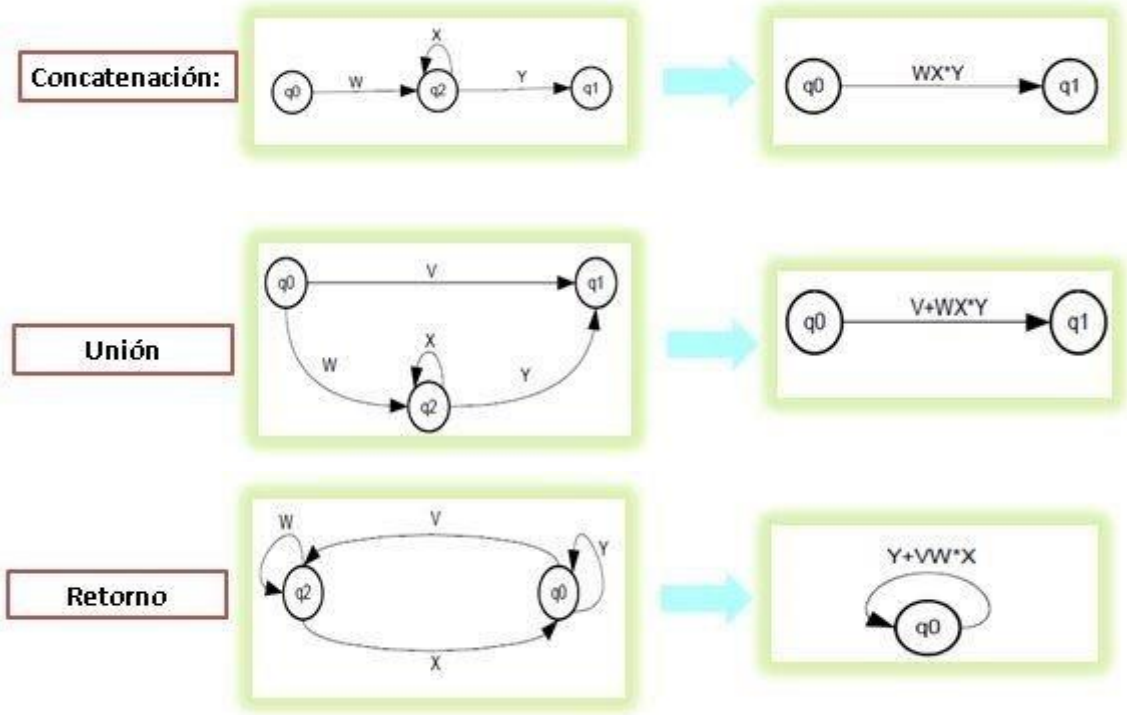
ACTIVIDAD SEMANA 10

Los lenguajes descritos por expresiones regulares son los lenguajes reconocidos por los autómatas finitos. Existe un algoritmo para convertir una expresión regular en el autómata finito no determinístico correspondiente. El algoritmo construye a partir de la expresión regular un autómata con transiciones vacías, es decir un autómata que contiene arcos rotulados con ε. Luego este autómata con transiciones vacías se puede convertir en un autómata finito sin transiciones vacías que reconoce el mismo lenguaje.

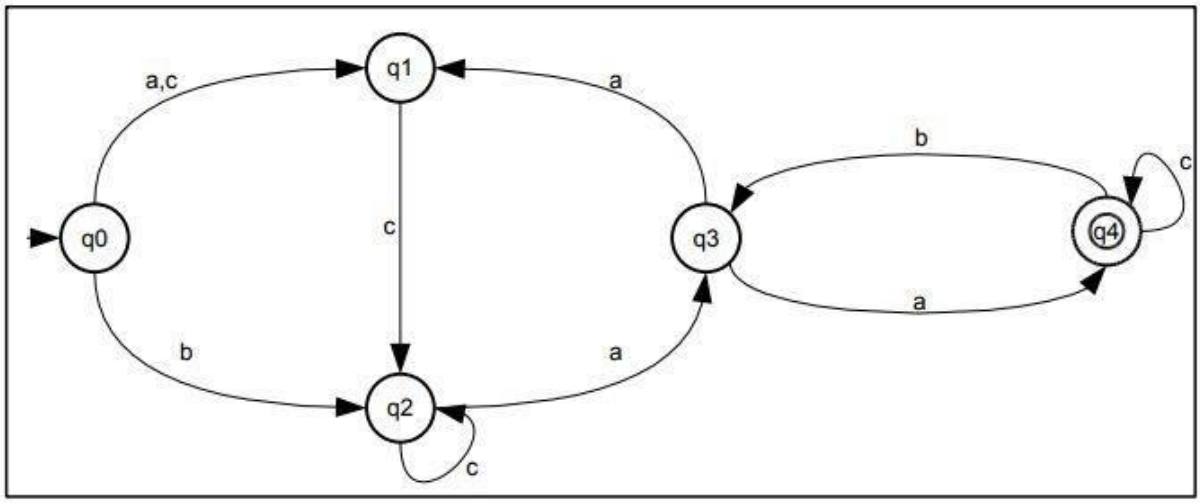
- Dada una expresión regular existe un autómata finito capaz de reconocer el lenguaje que ésta define.
- Recíprocamente, dado un autómata finito, se puede expresar mediante una expresión regular del lenguaje que reconoce.

Conversión de un AFD en una expresión regular mediante la eliminación de estados

En este texto vamos a ver uno de los métodos que se usan para transformar autómatas finitos deterministas en expresiones regulares, el método de eliminación de estados. Cuando tenemos un autómata finito, determinista o no determinista, podemos considerar que los símbolos que componen a sus transiciones son expresiones regulares. Cuando eliminamos un estado, tenemos que reemplazar todos los caminos que pasaban a través de él como transiciones directas que ahora se realizan con el ingreso de expresiones regulares, en vez de con símbolos. Los casos bases son los siguientes:

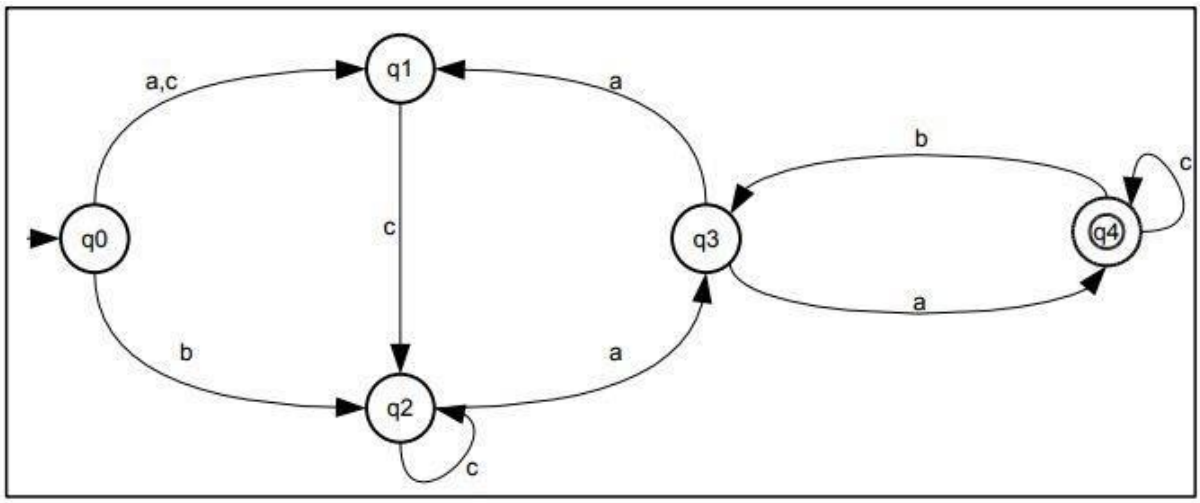


**Ejemplo**




1. Transformar todas aquellas transiciones que contemplan más de un símbolo en expresiones regulares del tipo “R+P.

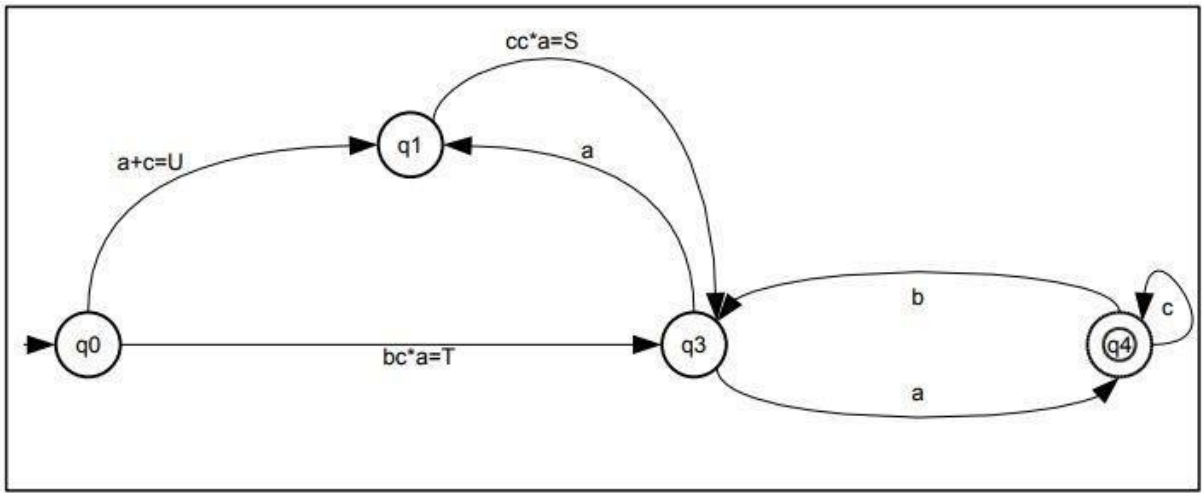
Por lo tanto, la transición q0 a q1 , queda como “a+c”. Todas las demás permanecen iguales, por ser símbolos.




Se procede a eliminar el estado q2 . Verificar todos los caminos que atraviesan q 2 :

- q 0 puede llegar a q3 pasando por q 2 .
- q1 puede llegar a q3 pasando por q 2 .

Al **eliminar es estado q2** las transiciones del AFD quedan:




1) Para llegar a q3 , q0 primero debe pasar por q2 usando el símbolo “b”, luego puede pasar repetidas v eces por q2 (o ninguna vez) usando cero, uno o varios símbolos “c” y, por último, pasa de q 2 a q 3 con el símbolo “a”. Esto queda expresado con la expresión regular T = bc*a

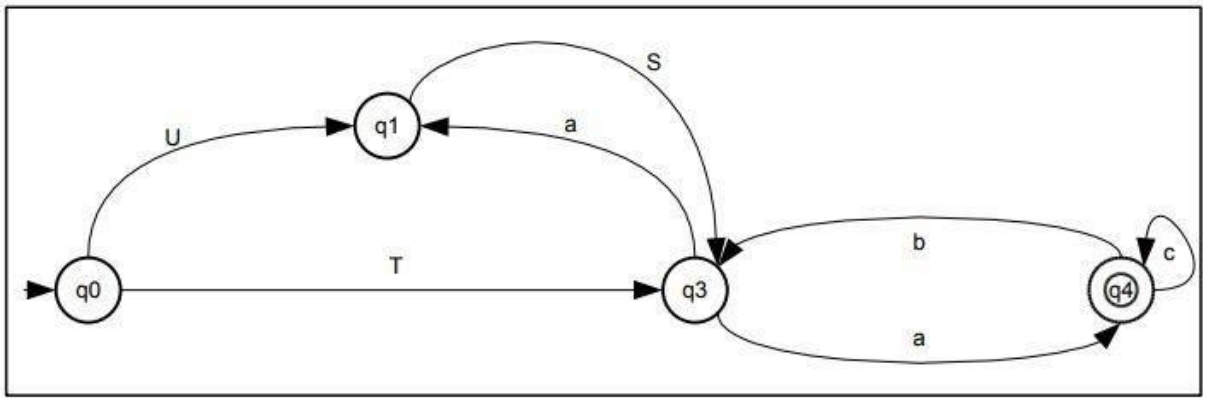
2) Para pasar de $q_1$ a $q_3$ , primero se debe llegar a $q_2$ con una “c”. Después de esto, se puede volver repetidas veces a $q_2$ , usando el símbolo “c”.

Se puede pasar cero, una o muchas veces por q2

de esta forma.

Como último paso, se debe llegar desde q2 a q3 con una “a”. Al concatenar estos tres caminos se obtiene:

S = cc*a



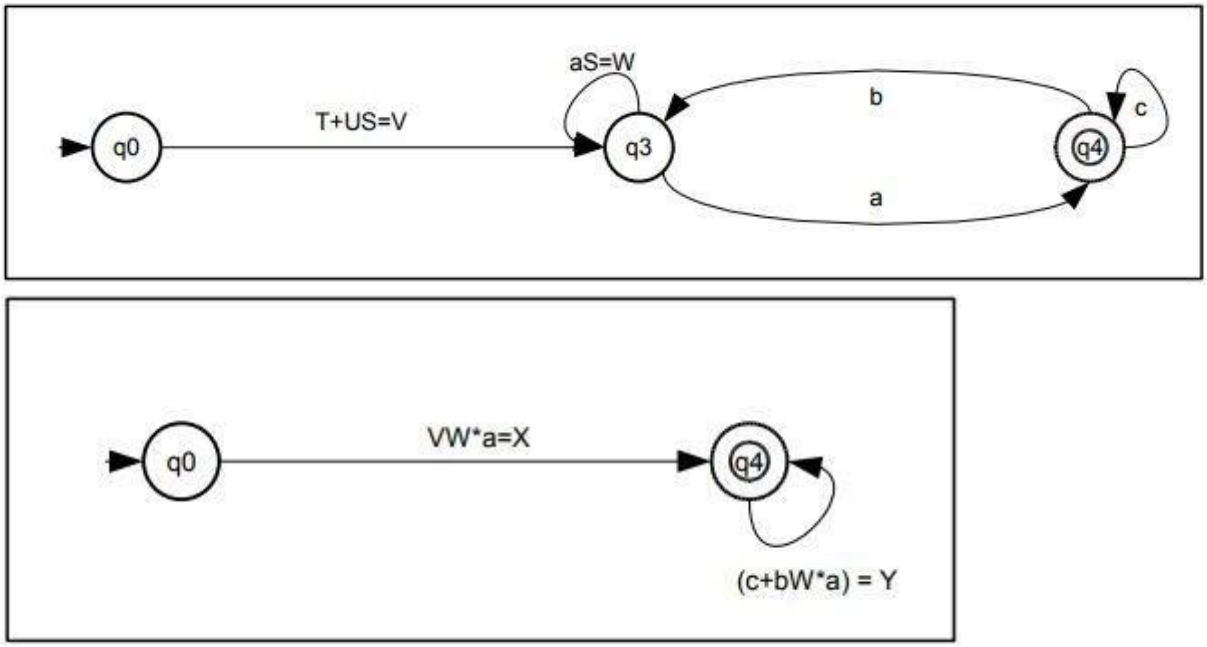

Al hacer las simplificaciones correspondiente, sólo se dejan las expresiones regulares resumidas por las letras mayúsculas (incluyendo a+c = U), y el autómata queda:

**Eliminar q1 .** Las transiciones afectadas son:

1) De q0 a q3 , que puede hacerlo a través de q1 o directamente, usando T. La expresión regular que va de q0 a q3 a través de q1 es US y la expresión que va directamente de q0 a q3 es T. Como puede ser una o la otra, como resultado queda la siguiente expresión: V = T + US

2) De q3 a q3 , a través de q1 . Esto es: Primero llega a q1 con una “a”, y luego pasa de q1 a q3 , usando S. La expresión resultante es:

W = aS



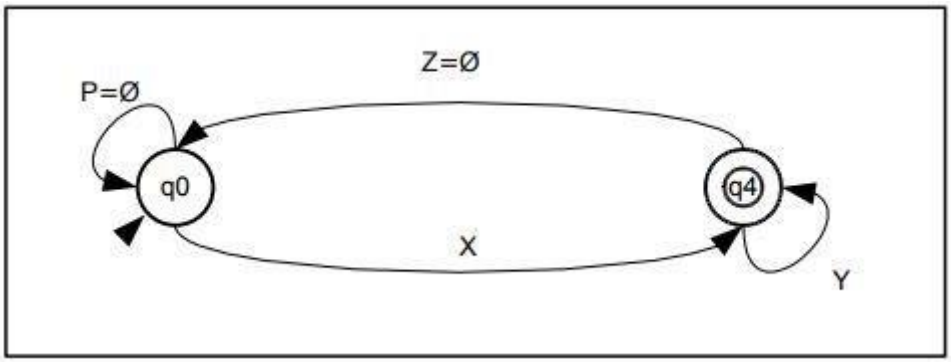

**Eliminar q3** . Las transiciones afectadas son:

1) De q0 a q4 , pasando por q3 : Primero va de q0 a q3 usando la expresión V. Luego de q3 a q3 repetidas veces, usando W. Por último, pasa de q3 a q4 usando una “a”. La expresión regular queda : X = V.W*.a

2) De q4 a q4 , pasando por q3 o directamente por q4 con una “c”. Para pasar a través de q3 , primero va de q4 a q 3

con una “b”. Luego de q3 a q 3 usando W. Y, por último, pasa de q 3 a q 4 con una “a”. Como puede ir desde q4 a q4 usando la “c” o través de q3 , la expresión regular que queda es: Y = (c + bW*a)


Al dejar las Expresiones Regulares de forma resumida, el último autómata queda:

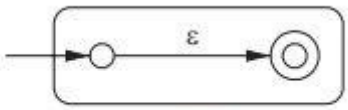



Completando las transiciones vacías (de q4 a q1 y de q1 a q1 ) con las expresiones regulares vacías Z y P

De esta forma, se puede generar la expresión regular final a partir de la fórmula (P + XY*Z)*XY*

**Expresión Final = (P + XY*Z)*XY* = (Ø* + XY*Ø)* XY* = (ε+ Ø)*XY* = ε *XY* = XY***

Conversión de una expresión regular a un autómata

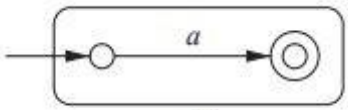



(a)

En la parte (a) vemos cómo se maneja la expresión ε. Puede verse fácilmente que el lenguaje del autómata es {ε}, ya que el único camino desde el estado inicial a un estado de aceptación está etiquetado con ε

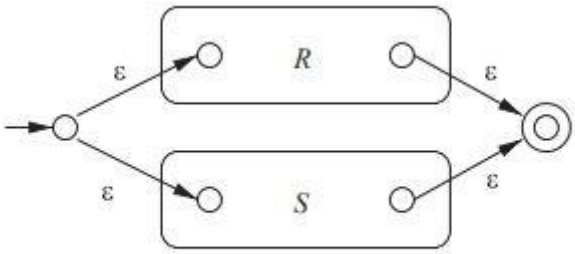
(b)

La parte (b) muestra la construcción de /0. Claramente, no existen caminos desde el estado inicial al de aceptación, por lo que /0 es el lenguaje de este autómata.



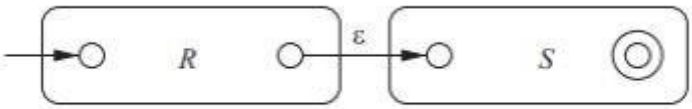

(c)

La parte (c) proporciona el autómata que reconoce una expresión regular a. Evidentemente, el lenguaje de este autómata consta de una cadena a, que es también L(a).



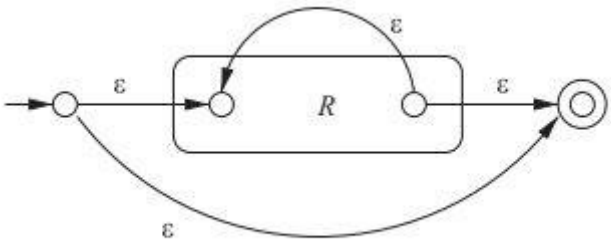

(a)

La expresión es de la forma R + S para dos expresiones R y S más pequeñas. Por tanto, el lenguaje del autómata de la (a) es L(R) ∪ L(S).




(b)

La expresión es de la forma RS para expresiones R y S más pequeñas. Por tanto, los caminos en el autómata de la (b) son todos y sólo los etiquetados con cadenas pertenecientes a L(R)L(S).



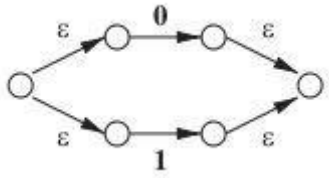

(c)

La expresión es de la forma R∗ para una expresión R más pequeña. Dicho camino acepta ε, que pertenece a L(R∗) sin importar qué expresión sea R.

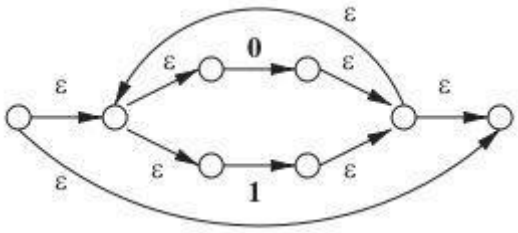
**Ejemplo**

Deseamos convertir la expresión regular (0+ 1)∗1(0 + 1) en un AFN-ε.

El **primer paso** consiste en construir un autómata para 0 + 1.




**Luego** Construiremos (0+1)* a partir de (0+1), Colocando una trición vacia en el nodo final de 0+1 devolviendome al inicio para las veces que se repita el (0+1), luego colocaremos un nodo anterior al nodo inicial de (0+1) y añadiremos una transición vacia entre este nodo y un nodo adelante del nodo final para cuando (0+1)^0




Finalmente colocaremos una transición 1 que conecte (0+1)* con (0+1) dando como resultado