UNIVERSIDAD AUTÓNOMA DE CHIAPAS

Facultad de Contaduría y Administración Campus I

NOMBRE DE LA MATERIA:

COMPILADORES

TEMA:

DEFINIR CONCEPTOS

NOMBRE DEL ALUMNO:

MARCO ANTONIO ESTRADA DE LA CRUZ

SEMESTRE Y GRUPO:

6° M

MATRICULA:

A210179

FECHA:

28 de Enero de 2024

DEFINIR EL CONCEPTO DE EXPRESIÓN REGULAR

Las expresiones regulares están estrechamente relacionadas con los autómatas finitos no deterministas y pueden considerarse una alternativa, que el usuario puede comprender fácilmente, a la notación de los AFN para describir componentes de software. Por tanto, las expresiones regulares sirven como lenguaje de entrada de muchos sistemas que procesan cadenas. Algunos ejemplos son los siguientes:

• Comandos de búsqueda tales como el comando grep de UNIX o comandos equivalentes para localizar cadenas en los exploradores web o en los sistemas de formateo de texto.

• Generadores de analizadores léxicos, como Lex o Flex. Recuerde que un analizador léxico es el componente de un compilador que divide el programa fuente en unidades lógicas o sintácticas formadas por uno o más caracteres que tienen un significado.

I.- Explicar los tipos de operadores de expresiones regulares.

Las expresiones regulares denotan lenguajes. Por ejemplo, la expresión regular $01*+10*$ define el lenguaje que consta de todas las cadenas que comienzan con un 0 seguido de cualquier número de 1s o que comienzan por un 1 seguido de cualquier número de 0s. No esperamos que el lector sepa ya cómo interpretar las expresiones regulares, por lo que por el momento tendrá que aceptar como un acto de fe nuestra afirmación acerca del lenguaje de esta expresión. Enseguida definiremos todos los símbolos empleados en esta expresión, de modo que pueda ver por qué nuestra interpretación de esta expresión regular es la correcta. Antes de describir la notación de las expresiones regulares, tenemos que estudiar las tres operaciones sobre los lenguajes que representan los operadores de las expresiones regulares.

Estas operaciones son:

1. La unión de dos lenguajes L y M, designada como L '∪ M, es el conjunto de cadenas que pertenecen a L, a M o a ambos. Por ejemplo, si L = {001,10,111} y M = {ε ,001}, entonces L ' ∪ M = {ε ,10,001,111}.

2. La concatenación de los lenguajes L y M es el conjunto de cadenas que se puede formar tomando cualquier cadena de L y concatenándola con cualquier cadena de M. Recuerde la Sección 1.5.2, donde definimos la concatenación de una pareja de cadenas; el resultado de la concatenación es una cadena seguida de la otra. Para designar la concatenación de lenguajes se emplea el punto o ningún operador en absoluto, aunque el operador de concatenación frecuentemente se llama “punto”. Por ejemplo, si

L={001,10,111} y M = {ε ,001}, entonces L.M, o simplemente LM, es {001,10,111,001001,10001,111001}. Las tres primeras cadenas de LM son las cadenas de L concatenadas con ε. Puesto que ε es el elemento identidad para la concatenación, las cadenas resultantes son las mismas cadenas de L. Sin embargo, las tres últimas cadenas de LM se forman tomando cada una de las cadenas de L y concatenándolas con la segunda cadena de M, que es 001. Por ejemplo, la concatenación de la cadena 10 de L con la cadena 001 de M nos proporciona la cadena 10001 para LM.

3. La clausura (o asterisco, o clausura de Kleene) 1 de un lenguaje L se designa mediante L' ∗ y representa el conjunto de cadenas que se pueden formar tomando cualquier número de cadenas de L, posiblemente con repeticiones (es decir, la misma cadena se puede seleccionar más de una vez) y concatenando todas ellas. Por ejemplo, si L = {0,1}, entonces L' ∗ es igual a todas las cadenas de 0s y 1s. Si L = {0,11}, entonces L' ∗ constará de aquellas cadenas de 0s y 1s tales que los 1s aparezcan por parejas, como por ejemplo 011, 11110 y ε , pero no 01011 ni 101. Más formalmente, L' ∗ es la unión infinita $\cup i \geq 0$ Li, donde L0 = {ε }, L1 = L y Li, para i > 1 es LL· · ·L (la concatenación de i copias de L).

EJEMPLO:

Operador	Descripción	Ejemplo	Devuelve
^	Coincide con el principio de una cadena	^abc	abc, abcdef..., abc123
$	Coincide con el final de una cadena	abc$	mi:abc, 123abc, theabc
.	Coincide con cualquier carácter como comodín	a.c	abc, asc, a123c
|	Un carácter O	abc|xyz	abc o xyz
(...)	Captura los valores entre paréntesis.	(a)b(c)	a y c
[...]	Coincide con todo lo que esté entre corchetes	[abc]	a, b, o c
[a-z]	Coincide con los caracteres en minúscula entre a y z	[b-z]	bc, mente, xyz

II.- Explicar el proceso de conversión de DFA a expresiones regulares.

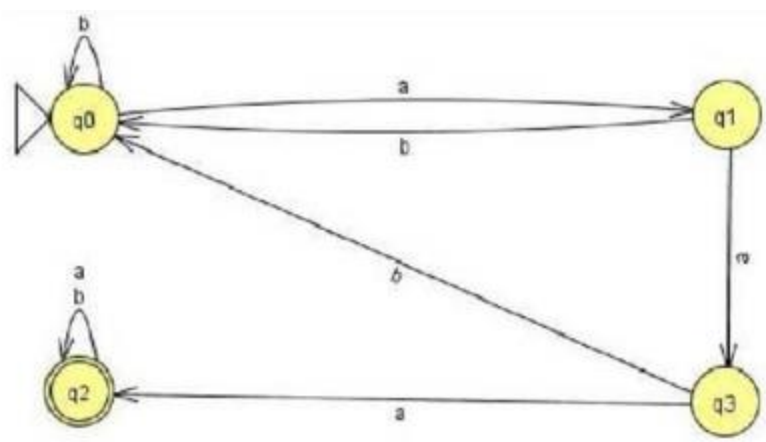
MÉTODO DE LA ELIMINACIÓN DE ESTADOS

Este método consiste en seleccionar tres estados: qr, el cual no deberá ser ni el estado inicial, ni ninguno de los estados finales o de aceptación, también se deberá seleccionar un estado qz y qy, de manera que qz pueda llegar (por medio de transiciones) a qy utilizando a qr como estado intermedio entre estos. Después de haber seleccionado estos estados, se debe proceder a eliminar el estado qr, haciendo una transición que vaya de qz a qy y que por medio de la concatenación de las transiciones que llegan de qz a qr y salen de qr a qy (incluyendo las que hacen un bucle en qr). En caso de que ya exista una transición que va de qz a qy, se hace la unión de la Expresión Regular de dicha transición con la Expresión Regular de la nueva transición antes creada. Esto se repite hasta que solo existan estados iniciales y finales en el DFA. Luego de tener la máquina de esta forma se debe generar la Expresión Regular a partir de ella.

Para un mejor entendimiento de la eliminación de los estados utilizaremos el siguiente

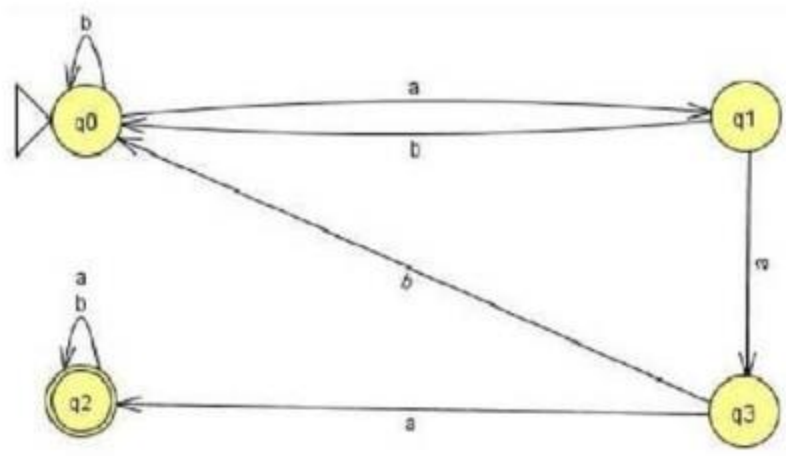
ejemplo:

Haremos la conversión del siguiente DFA a una Expresión Regular:



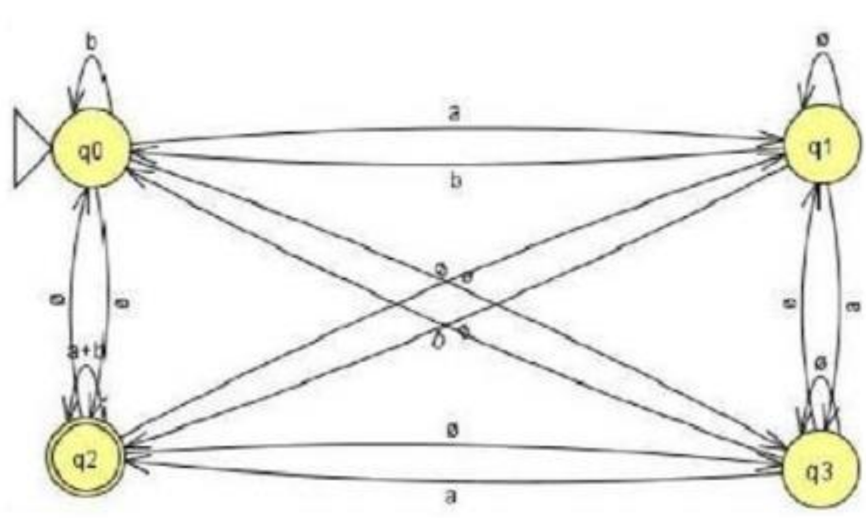
PASO 1: Por cada transición Qi3 que pueda ser recorrida con múltiples símbolos, se hará una transición Qj(siendo esta una transición que contiene una Expresión Regular)4 que contenga los símbolos de dicha transición Qi representados como una Expresión Regular, específicamente como una unión.

APLICACIÓN: Como podemos observar, la transición que hace un bucle en q2 es la única transición que tiene múltiples símbolos con la cual puede ser transitada, por lo tanto la representaremos como una unión de la siguiente manera: a + b. Nos resulta en:



PASO 2: Por cada estado qi, se debe verificar si hay una transición Qj que llegue a cada estado qn (donde qn = qi) de la máquina. En caso de no existir esta transición se deberá agregar una transición que va desde qi hasta qn con el valor ∅.

APLICACIÓN: Al aplicar el paso 2 a nuestro DFA, podemos ver que no hay transición de q0 a q2, tampoco existe un bucle en q1, tampoco hay transición de q3 a q2, etc. Por lo tanto agregaremos todas las transiciones que hacen falta para conectar cada estado con el resto de estados de la máquina. Estos estados tendrán el símbolo ø. Por lo tanto nuestro DFA resulta en:



PASO 3: Seleccionar un estado qr, talque qr NO sea un estado inicial y/o final. Luego, por cada estado qz se selecciona un camino, pasando por qr, hacia cada estado qy del DFA, talque qz ≠ qr y qy ≠ qr. Ahora se crea una transición Qj que tenga como Expresión Regular el símbolo (o Expresión Regular) de la transición que va de qz a qr concatenado con el símbolo de la transición que va de qr a qy. Al bucle que se hace en qr se le aplicará la operación de clausura (o clausura Kleene) y se concatenará con la Expresión Regular antes encontrada. A esta nueva transición Qj se le aplica una operación de unión con el símbolo de la transición que va de qz a qy. La transición Qj deberá quedar de la forma Ti S∗ Tj + Tk. Esta nueva transición Qj transitará del estado q al estado q .

APLICACIÓN: Ahora bien, seleccionaremos como estado qr a q1. Haciendo la concatenación da cada transición desde todos los estados qz hacia todos los estados qy usando a qr como intermediario, nos resulta las siguientes Expresiones Regulares:

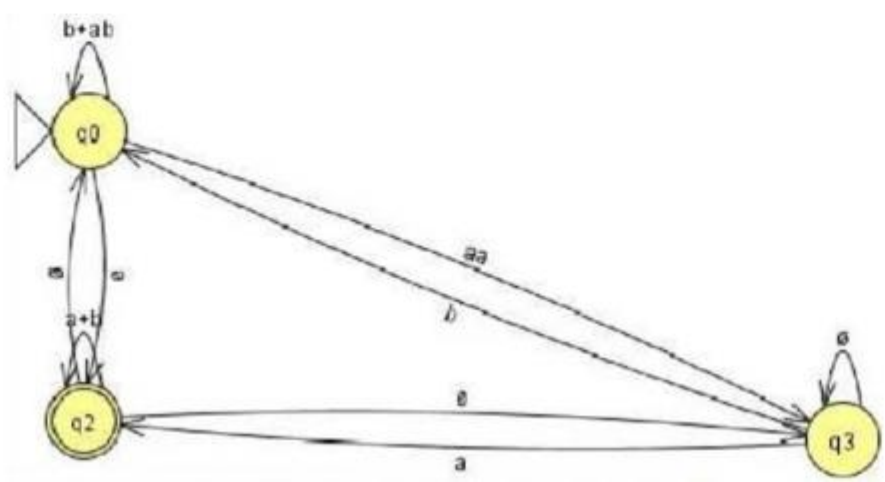
q_x	q_y	O_j
0	0	ab
0	2	aØ
0	3	aa
2	0	bØ
2	2	bØ
2	3	aØ
3	0	bØ
3	2	bØ
3	3	aØ

Ahora le aplicaremos la operación de clausura al bucle de qr y haremos la concatenación con el Qj que ya encontramos. Resulta en:

qx	qy	O
0	0	aØ*b
0	2	aØ*Ø
0	3	aØ*a
2	0	bØ*Ø
2	2	bØ*Ø
2	3	aØ*Ø
3	0	bØ*Ø
3	2	bØ*Ø
3	3	aØ*Ø

El siguiente paso es hacer la unión del Qj que ya tenemos con el símbolo de la transición que va de qz a qy directamente. También agregaremos una columna con la Expresión Regular ya simplificada, por lo tanto obtenemos:

Excelente! Hemos logrado eliminar el estado q1 de nuestro DFA. Ahora se deben seguir los mismos pasos hasta tener un DFA que solo contenga el estado inicial y los finales (en este caso q0 y q2). El DFA nos queda de la siguiente manera:



Ahora que ya hemos comprendido los pasos esenciales de la eliminación de estados, presentaremos la tabla para eliminar el estado q3:

q_x	q_y	O_j	Simplificación
0	0	$aa\emptyset^*b + ab$	$aab + ab + b$
0	2	$+ baa\emptyset^*a + \emptyset$	aaa
2	0	$\emptyset\emptyset^*a + \emptyset$	$\emptyset$
2	2	$\emptyset\emptyset^*a + b + a$	$b + a$

Hemos terminado de eliminar los estados no iníciales y no finales de nuestro DFA. Aplicando todas las Expresiones Regulares de la tabla anterior a nuestro DFA, nos quedaría así:

PASO 4: Si tenemos un estado inicial qz y un estado final qy donde qz ≠ qy, se debería generar una Expresión Regular a partir de este DFA de la forma $(R + SU^*T)^*SU^*$, donde R es un bucle en qz, S es el camino que va de qz a qy, U es un bucle en q2 y T es el camino que va de qy a qz.

APLICACIÓN: Primero identificamos las Expresiones Regulares R, S, U y T. Tenemos que R = b + ab + aab, S = aaa, U = a + b y T = ∅. Ahora que ya tenemos los valores, procedemos a hacer la Expresión Regular final:

$ER = (b + ab + aab + (aaa)(a + b)^*\emptyset)^*(aaa)(a + b)^* = (b + ab + aab)^*(aaa)(a+b)^*$