UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN “CAMPUS I”

LICENCIATURA EN INGENIERÍA EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6“M”

ALUMNO:

LOPEZ RAMIREZ MARTI HERNAN

DOCENTE:

Dr. Luis Alfaro Gutiérrez

TAREA:

Define los siguientes conceptos y realizar los ejercicios. -

Actividad I, II.

TUXTLA GUTIÉRREZ, CHIAPAS A 28 de enero de 2024

INDICE

Definir el concepto de expresión regular.

DEFINICION: Una expresión regular constituye una forma de representar un lenguaje en forma sintética. No cualquier lenguaje se puede representar mediante una expresión regular, los que son representables mediante una expresión regular se denominan lenguajes regulares. Dada una expresión regular r el lenguaje que representa es L(r) Una expresión regular se construye a partir de expresiones regulares más simples, usando un conjunto de reglas definitorias.

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

Operadores comunes

Para definir patrones de coincidencia, puede utilizar estos operadores comunes:

<table>
<tr><th>Operador</th><th>Descripción</th><th>Ejemplo</th><th>Devuelve</th></tr>
<tr><td>^</td><td>Coincide con el principio de una cadena</td><td>^abc</td><td>abc, abcdef..., abc123</td></tr>
<tr><td>$</td><td>Coincide con el final de una cadena</td><td>abc$</td><td>mi:abc, 123abc, theabc</td></tr>
<tr><td>.</td><td>Coincide con cualquier carácter como comodín</td><td>a.c</td><td>abc, asc, a123c</td></tr>
<tr><td>|</td><td>Un carácter O</td><td>abc|xyz</td><td>abc o xyz</td></tr>
<tr><td>(...)</td><td>Captura los valores entre paréntesis.</td><td>(a)b(c)</td><td>a y c</td></tr>
<tr><td>[...]</td><td>Coincide con todo lo que esté entre corchetes</td><td>[abc]</td><td>a, b, o c</td></tr>
<tr><td>[a-z]</td><td>Coincide con los caracteres en minúscula entre a y z</td><td>[b-z]</td><td>bc, mente, xyz</td></tr>
<tr><td>[0-9]</td><td>Coincide con cualquier valor numérico entre 0 y 9.</td><td>[0-3]</td><td>3201</td></tr>
<tr><td>{x}</td><td>El número exacto de veces que debe coincidir</td><td>(abc){2}</td><td>abcabc</td></tr>
<tr><td>{x,}</td><td>El número mínimo de veces que debe coincidir</td><td>(abc){2,}</td><td>abcabcabc</td></tr>
<tr><td>*</td><td>Coincide con cualquier cosa en lugar de *, o una coincidencia "codiciosa".</td><td>ab*c</td><td>abc, abbcc, abcdc</td></tr>
<tr><td>+</td><td>Coincide con el carácter anterior al + una o más veces</td><td>a+c</td><td>ac, aac, aaac</td></tr>
<tr><td>?</td><td>Coincide con el carácter anterior a ? cero o una vez, o una coincidencia "no codiciosa".</td><td>ab?c</td><td>ac, abc</td></tr>
<tr><td>/</td><td>Escapa el carácter después de /, O crea una secuencia de escape</td><td>a/bc</td><td>a c, con el espacio correspondiente a /b</td></tr>
</table>

Para utilizar el carácter literal de un operador dentro de un patrón, **no** como regex:

- Para un circunflejo (`^`), punto (`.`), corchete abierto (`[`), signo del dólar (`$`), paréntesis abierto o cerrado (`(`) o (`)`), tubo (`|`), asterisco (`*`), signo más (`+`), signo de interrogación (`?`), llave abierta (`{`), o barra invertida (`\`), siga con el operador de escape (`\`).
- Para un corchete final (`]`) o una llave final (`}`), conviértalo en el primer carácter, con o sin apertura `^`.
- Para un guión (`-`), conviértalo en el primer o último carácter, o en el segundo punto final de un rango.

Coincide con el inicio o el final de la cadena (`^` y `$`)

- Para hacer coincidir patrones al principio o al final de la cadena, utilice los operadores ^ y $, respectivamente. Por ejemplo:

Ejemplo	Partidos
`^El`	Cualquier cadena que empiece por `El`
`de desesperación$`	Cualquier cadena que termine con `of despair`
`^abc$`	Una cadena que empieza y termina con `abc`-una coincidencia exacta

Caracteres coincidentes (`*`, `+`, y `?`)

Para hacer coincidir patrones basados en un carácter específico, siga el carácter con el operador `*`, `+`, o `?`. Estos operadores indican el número de veces que debe aparecer el carácter para obtener una coincidencia: cero o más, uno o más, o uno o cero, respectivamente. Por ejemplo:

Ejemplo	Partidos
`ab*`	Una cadena que contiene `a`, seguida de cero o más `b`s-`ac`, `abc`, o `abbc`
`ab+`	Una cadena que contiene `a`, seguido de uno o más `b`s-`abc` o `abbc`, pero **no** `ac`
`¿Ab?`	Una cadena que contiene `a`, seguido de cero o uno `b`s-`ac` o `abc`, pero **no** `abc`
`a?b+$`	Una cadena que termina con uno o más `b`s, con o sin un `a` precedente ; por ejemplo, `ab`, `abb`, `b`, o `bb`, pero **no** `aab` o `aabb`

Frecuencia de coincidencia de caracteres (`{ . . . }` o `( . . . )`)

Para buscar un patrón basado en la frecuencia de aparición de un único carácter, escriba a continuación el número o el intervalo de casos, entre llaves (`{...}`). Por ejemplo:

Ejemplo	Partidos
`ab{2}`	Una cadena que contiene `a`, seguida de exactamente 2 `b`s-`abb`
`ab{2,}`	Cadena que contiene `a`, seguida de al menos 2 `b`s-`abb`, `abbbb`, etc.
`ab{3,5}`	Una cadena que contiene `a`, seguida de tres a cinco `b`s-`abbb`, `abbbb`, o `abbbbb`

Coincidencia de uno de varios patrones (`|`)

Para que coincida con uno de varios patrones -como `este` O `ese`- utilice el operador O `|`. Por ejemplo:

Ejemplo	Partidos
`hola`	Una cadena que contiene `hi` o `hola`
`(b|cd)ef`	Una cadena que contiene `bef` o `cdef`
`(a|b)*c`	Una cadena que tiene una secuencia alternada de `a`s y `b`s, terminando con `c`

Coincide con cualquier carácter (`.`)

Para representar cualquier carácter en un patrón a comparar, utilice el operador comodín `.`. Por ejemplo:

Ejemplo	Partidos
`a.[0-9]`	Una cadena que contiene `un`, seguido de cualquier carácter y un dígito
`^.{3}$`	Cualquier cadena de exactamente tres caracteres

Coincidencia de posición de caracteres (`[ . . . ]`)

Para hacer coincidir un patrón basado en la posición de un carácter, utilice paréntesis (`[...]`). Por ejemplo:

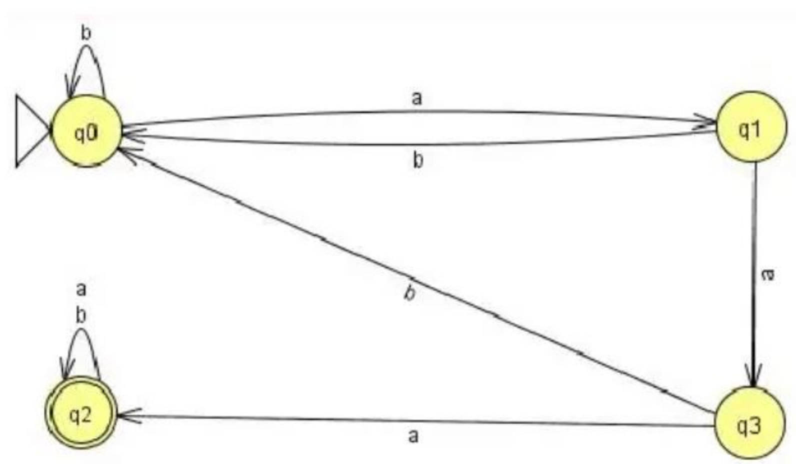
Ejemplo	Partidos
`[ab]`	Una cadena que contiene `a` o `b`; equivalente a `a|b`
`[a-d]`	Cadena que contiene una minúscula `a`, `b`, `c`, o `d`; equivalente a `a|b|c|d` o `[abcd].`
`^[a-zA-Z]`	Una cadena que empieza por cualquier letra, independientemente de mayúsculas y minúsculas
`[0-9]%`	Una cadena que contiene cualquier dígito seguido de un signo de porcentaje
`,[a-zA-Z0-9]$`	Una cadena que termina con una coma seguida de cualquier carácter

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

METODO DE ELIMINACION

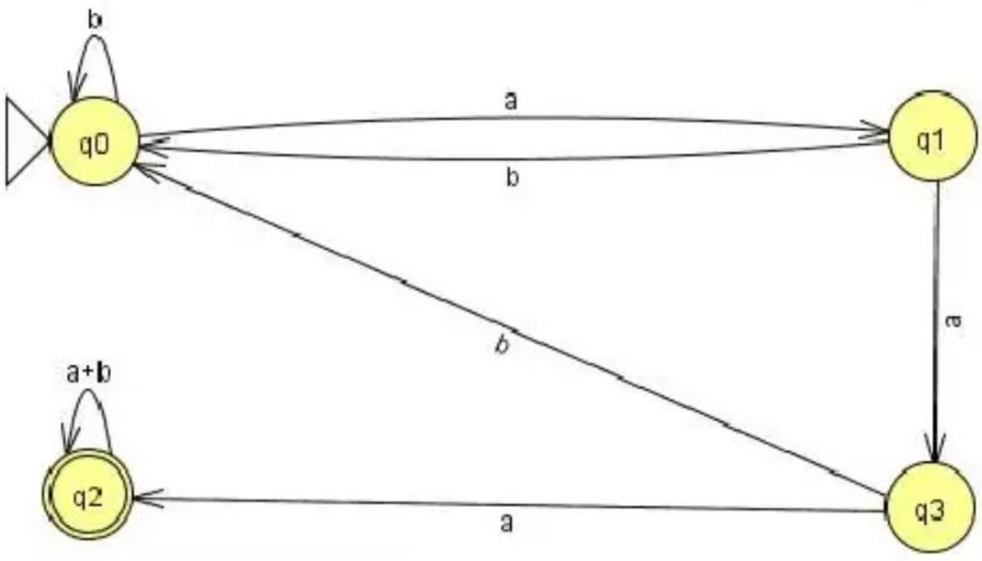
Básicamente este método consiste en seleccionar tres estados: qR el cual no deberá ser ni el estado inicial, ni ninguno de los estados finales o de aceptación, también se deberá seleccionar un estado qx y qy, de manera que qx pueda llegar (por medio de transiciones) a qy utilizando a qr, como estado intermedio entre estos. Después de haber seleccionado estos estados, se debe proceder a eliminar el estado qr, haciendo una transición que vaya de q a qy y que por medio de la concatenación de las transiciones que llegan de qx a qy salen de q, a qy (incluyendo las que hacen un bucle en q,). En caso de que ya exista una transición que va de qx 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.

Haremos la conversión del siguiente DFA a una Expresión Regular:



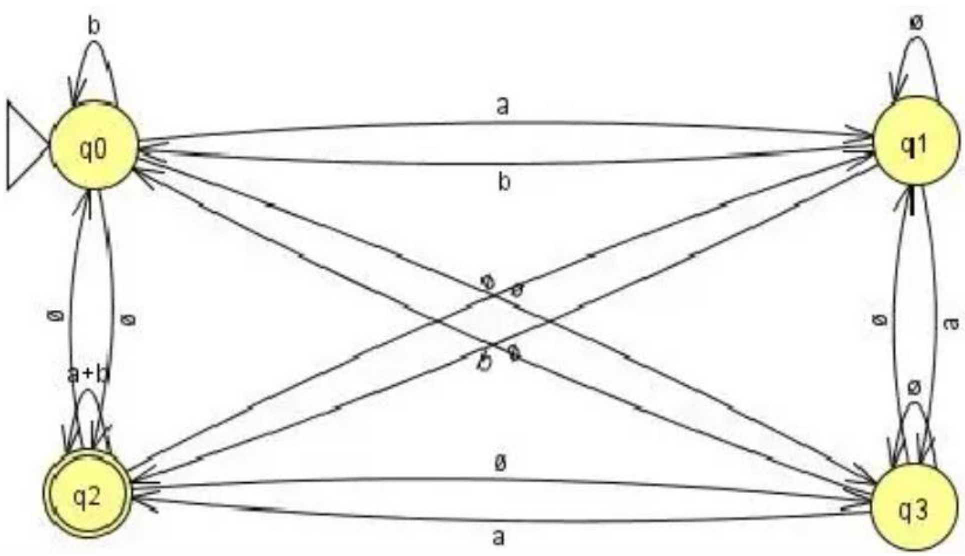
PASO 1: Por cada transición Q_i[3] que pueda ser recorrida con múltiples símbolos, se hará una transición Q_j(siendo esta una transición que contiene una Expresión Regular)[4] que contenga los símbolos de dicha transición Q_i representados como una Expresión Regular, específicamente como una unión.

APLICACIÓN: Como podemos observar, la transición que hace un bucle en q_2 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 q_i, se debe verificar si hay una transición Q_j que llegue a cada estado q_n (donde $q_n = q_i$) de la máquina. En caso de no existir esta transición se deberá agregar una transición que va desde q_i hasta q_n con el valor Ø.

APLICACIÓN: Al aplicar el paso 2 a nuestro DFA, podemos ver que no hay transición de q_0 a q_2, tampoco existe un bucle en q_1, tampoco hay transición de q_3 a q_2, 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:



ILUSTRACION 3: PASO 2

PASO 3: Seleccionar un estado q_r, talque q_r NO sea un estado inicial y/o final. Luego, por cada estado q_x se selecciona un camino, pasando por q_r, hacia cada estado q_y del DFA, talque $q_x \neq q_r$ y $q_y \neq q_r$. Ahora se crea una transición Q_j que tenga como Expresión Regular el símbolo (o Expresión Regular) de la transición que va de q_x a q_r concatenado con el símbolo de la transición que va de q_r a q_y. Al bucle que se hace en q_r se le aplicará la operación de clausura (o clausura Kleene) y se concatenará con la Expresión Regular antes encontrada. A esta nueva transición Q_j se le aplica una operación de unión con el símbolo de la transición que va de q_x a q_y. La transición Q_j deberá quedar de la forma $T_i S^* T_j + T_k$. Esta nueva transición Q_j transitará del estado q_x al estado q_y.

APLICACIÓN: Ahora bien, seleccionaremos como estado q_r a q_1. Haciendo la concatenación da cada transición desde todos los estados q_x hacia todos los estados q_y usando a q_r como intermediario, nos resulta las siguientes Expresiones Regulares:

q_x	q_y	Q_j
0	0	ab
0	2	$a\emptyset$
0	3	aa
2	0	$b\emptyset$
2	2	$b\emptyset$
2	3	$a\emptyset$
3	0	$b\emptyset$
3	2	$b\emptyset$
3	3	$a\emptyset$

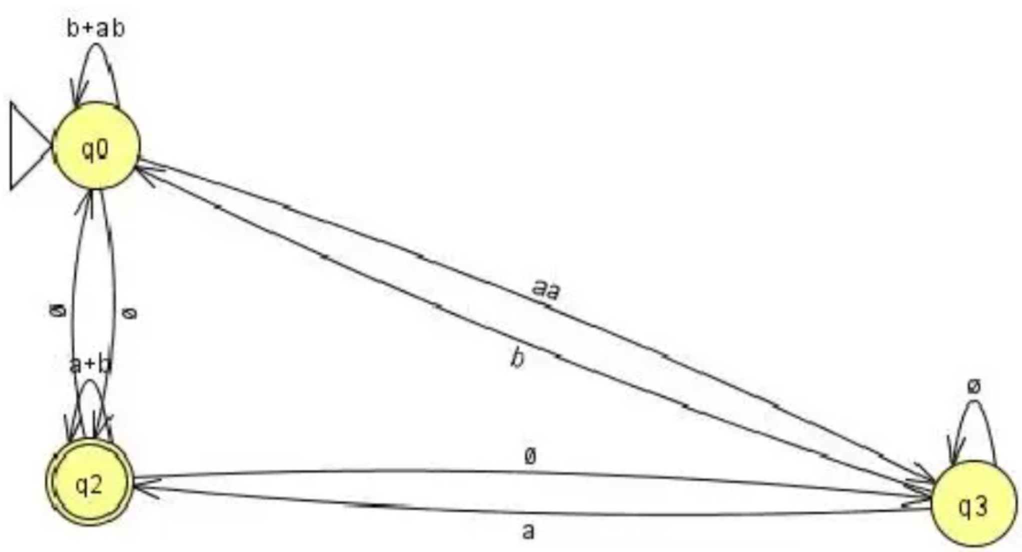
Ahora le aplicaremos la operación de clausura al bucle de q_r y haremos la concatenación con el Q_j que ya encontramos. Resulta en:

q_x	q_y	Q_j
0	0	$a\emptyset^* b$
0	2	$a\emptyset^*\emptyset$
0	3	$a\emptyset^* a$
2	0	$b\emptyset^*\emptyset$
2	2	$b\emptyset^*\emptyset$
2	3	$a\emptyset^*\emptyset$
3	0	$b\emptyset^*\emptyset$
3	2	$b\emptyset^*\emptyset$
3	3	$a\emptyset^*\emptyset$

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

q_x	q_y	Q_j	Simplificación
0	0	$a\emptyset^*b + b$	$\boldsymbol{ab + b}$
0	2	$a\emptyset^*\emptyset + \emptyset$	$\emptyset$
0	3	$a\emptyset^*a + \emptyset$	$\boldsymbol{aa}$
2	0	$b\emptyset^*\emptyset + \emptyset$	$\emptyset$
2	2	$b\emptyset^*\emptyset + a + b$	$\boldsymbol{a + b}$
2	3	$a\emptyset^*\emptyset + \emptyset$	$\emptyset$
3	0	$b\emptyset^*\emptyset + b$	b
3	2	$b\emptyset^*\emptyset + a$	a
3	3	$a\emptyset^*\emptyset + \emptyset$	$\emptyset$

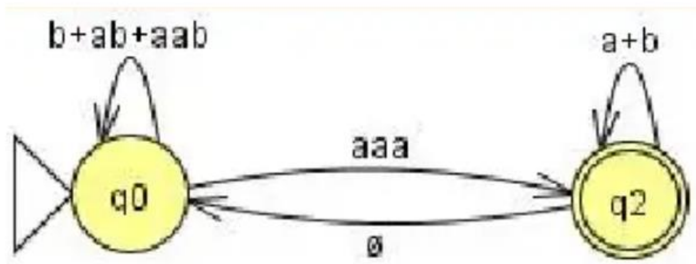
Excelente! Hemos logrado eliminar el estado q_1 de nuestro DFA. Ahora se deben seguir los mismos pasos hasta tener un DFA que solo contenga el estado inicial y los finales (en este caso q_0 y q_2). El DFA nos queda de la siguiente manera:



Ahora que ya hemos comprendido los pasos esenciales de la eliminación de estados, presentaremos la tabla para eliminar el estado q_3:

q_x	q_y	Q_j	Simplificación
0	0	$aa\emptyset^*b + ab + b$	$aab + ab + b$
0	2	$aa\emptyset^*a + \emptyset$	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 q_x y un estado final q_y donde $q_x \neq q_y$, se debería generar una Expresión Regular a partir de este DFA de la forma $(R + SU^*T)^*SU^*$, donde R es un bucle en q_x, S es el camino que va de q_x a q_y, U es un bucle en q_2 y T es el camino que va de q_y a q_x.

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

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

$$= (\boldsymbol{b} + \boldsymbol{ab} + \boldsymbol{aab})^*(\boldsymbol{aaa})(\boldsymbol{a} + \boldsymbol{b})^*$$

III.- Explicar leyes algebraicas de expresiones regulares

Leyes Distributivas

• Como la concatenacion no es conmutativa, tenemos ´ dos formas de la ley distributiva para la concatenacion: ´

• Ley Distributiva Izquierda para la concatenacion sobre ´ union: ´ L(M + N) = LM + LN

• Ley Distributiva Derecha para la concatenacion sobre ´ union: ´ (M + N)L = ML + NL

.

Ley de Idempotencia

• Se dice que un operador es idempotente (idempotent) si el resultado de aplicarlo a dos argumentos con el mismo valor es el mismo valor

• En general la suma no es idempotente: x + x 6= x (aunque para algunos valores s´ı aplica como 0 + 0 = 0)

• En general la multiplicacion tampoco es idempotente: ´ x × x 6= x

• La union e intersecci ´ on son ejemplos comunes de ´ operadores idempotentes. Ley idempotente para la union: ´ L + L = L

FUENTES

FernandoEscher. (s. f.). *DFA a expresion regular*. Scribd. https://es.scribd.com/doc/12929632/DFA-a-Expresion-Regular?doc_id=12929632&order=626455370

Expresiones Regulares. (2016). SSYL. https://www.frro.utn.edu.ar/repositorio/catedras/sistemas/2_anio/sintaxis/SSyL-cap3_2015_Expresiones%20Regulares.pdf

Expresiones Regulares. (2015b, mayo 6). inaoep. https://ccc.inaoep.mx/ingreso/automatas/expresionesRegulares.pdf