Universidad Autónoma de Chiapas | Campus 01 | Facultad de contaduría y administración.

Docente: Gutiérrez Alfaro Luis, Dr.

Materia: Taller de desarrollo 4.

Nombre del alumno (s):

- Carrasco Zavala Carlos Emmanuel | A210731.

Semestre: 6° | Grupo: “M”.

Conceptos. | AP .

Número de Actividad: Define los siguientes conceptos y ejercicios.-

Act. I.

Unidad 1

Tuxtla Gutiérrez, Chiapas; a 28 de Enero del 2024

Una expresión regular es una secuencia de caracteres que forma un patrón de búsqueda. Se utiliza para encontrar o reemplazar cadenas de texto que coinciden con ese patrón.

Ejemplo, la expresión regular **a*b** se corresponde con cualquier cadena que empiece con cero o más a y termine con una **b**.

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

Algunos de los operadores más comunes son:

Operador	Descripción	Ejemplo
Operador de coincidencia con cualquier carácter (.)	El carácter de punto representa a este operador.	**a.b** coincide con cualquier cadena de tres caracteres que empiece por **a** y termine por **b.**
Operador Match-zero-or-more (*)	Este operador repite la expresión regular anterior más pequeña posible tantas veces como sea necesario (incluido cero) para que coincida con el patrón	**a*** coincide con cualquier cadena compuesta por cero o más **a**. En otro ejemplo, **fo*** tiene una **o** repetitiva, no una **fo** repetitiva. Por lo tanto, **fo*** coincide con **f, fo, foo,** etc.
Operador de una o más coincidencias (+)	Este operador es similar al operador de cero o más coincidencias, con la excepción de que repite la expresión regular anterior al menos una vez.	**ca+r** coincide con **car** y con **caaaar**, pero no con **cr**
Operador de cero o una coincidencia (?)	Este operador es similar al operador de cero o más coincidencias, con la excepción de que repite la expresión	**ca?r** coincide con **car** y con **cr**, pero con nada más.

	regular anterior una vez o no la repite.	
Negar (^)	Niega una expresión.	**^a** coincide con cualquier carácter, excepto con **a**
Operadores de agrupación ((...))	La expresión regular trata las expresiones dentro de un paréntesis al igual que las matemáticas y los lenguajes de programación tratan una expresión entre paréntesis como una unidad. Las expresiones se procesan antes que la expresión fuera del paréntesis.	**f(a|b)a** coincide con **faa** y con **fba,** lo que significa que la operación **a|b** se procesa antes que el resto.
Operador de alternancia (|)	Las alternancias coinciden con una de las opciones de expresiones regulares: si coloca uno o más caracteres que representan el operador de alternancia entre dos expresiones regulares a y b, el resultado coincide con la unión de las cadenas con las que coinciden a y b.	**foo|bar|quux** coincidiría con cualquiera de **foo**, **bar** o **quux** Como ejemplo adicional, **(** y **)** son los operadores de apertura y cierre de grupo, por lo que **fo(o|b)ar** coincidiría con **fooar o fobar**. Por el contrario, **foo|bar** coincidiría con **foo o bar.**
Operadores de lista ([...] y [^ ...])	Una lista coincidente coincide con un único carácter representado por uno de los elementos de la lista. Un elemento es un carácter, una	**[ab]** coincide con **a** o con **b. [ad]*** coincide con la cadena vacía y cualquier cadena compuesta solo por varias **a** y varias **d** en cualquier orden.

	expresión de clase de carácter o una expresión de rango. Las listas no coincidentes son similares a las listas coincidentes, excepto que coinciden con un único carácter no representado por uno de los elementos de la lista.	Como ejemplo de no coincidente, **[^ab]** coincide con cualquier carácter, excepto con **a** o **b**
Operador de rango (-)	Representa los caracteres que están entre dos elementos en la secuencia de intercalación actual.	**[a-f]** representa todos los caracteres de **a** a **f** inclusive.
Dígito (\d)	Coincide con cualquier carácter de dígito (0-9).	Igual que **[0-9**]
No dígito (\D)	Coincide con cualquier carácter que no sea un dígito (0-9).	Igual que **[^0-9]**
Escape (\\)	Hace que el siguiente carácter de la expresión signifique el propio carácter, pero no un operador.	**\\.** significa punto, no el operador de coincidencia con cualquier carácter.

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

El método de eliminación de estados.

Este método consiste en ir eliminando uno por uno los estados del DFA, excepto el inicial y el final, y reemplazar los arcos que los conectan por RE que representen todas las posibles cadenas que pasan por el estado eliminado. Al final, se obtiene una RE que acepta el mismo lenguaje que el DFA original.

- *Si eliminamos un estado **s**, todos los caminos que pasen a través de **s** ya no existirán en el autómata.*
- *Si no queremos cambiar el lenguaje del autómata, tenemos que incluir, sobre un arco que vaya directamente desde **q** hasta **p**,*
    - *las etiquetas de los caminos que vayan desde algún estado **q** al estado **p**, pasando por **s**.*
- *Dado que la etiqueta de este arco ahora puede implicar cadenas en lugar de simples símbolos, e incluso un número infinito de tales cadenas, no podemos simplemente enumerar las cadenas como una etiqueta. Afortunadamente, existe una forma simple y finita de representar todas esas cadenas que es una expresión regular.*
- *Esto nos lleva a considerar autómatas que tienen expresiones regulares como etiquetas.*
    - *El lenguaje de un autómata es la unión, para todos los caminos desde el estado **inicial** hasta un estado **aceptación**, del lenguaje formado mediante la concatenación de los lenguajes de las expresiones regulares a lo largo de dicho camino.*

Observe que esta regla es coherente con la definición de lenguaje de cualquier variedad de autómata de las consideradas hasta el momento.

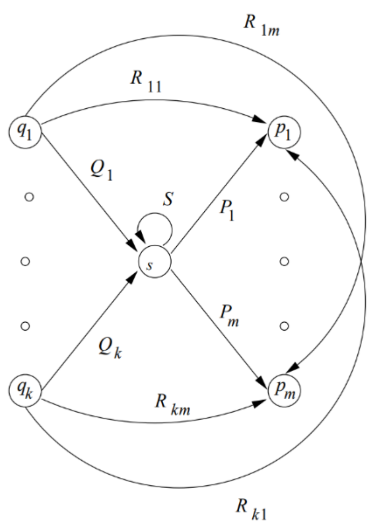
- *Cada símbolo **a**, o **ε** si está permitido, puede considerarse como una expresión regular cuyo lenguaje es una única cadena, **{a}** o **{ε}**.*

  Podemos considerar esta observación como la base del procedimiento de eliminación de estados que describimos a continuación.
  La Figura 1.1 muestra un estado genérico s que va a ser eliminado.
- *Supongamos que el autómata del que **s** es un estado tiene los estados predecesores **q1,q2,…,qk** y los estados sucesores **p1, p2,…, pm**.*

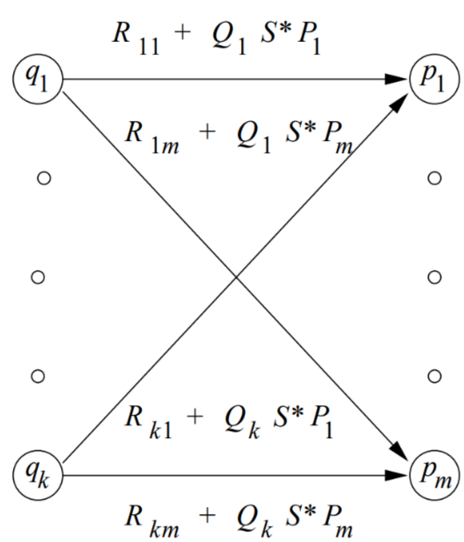
- *Es posible que alguno de los estados **q** sean también estados **p**, pero suponemos que **s** no está entre los **q** ni entre los **p**, incluso aunque exista un bucle de **s** a sí mismo, como se sugiere en la Figura 1.1.*

- *También se indica una expresión regular sobre cada arco que va desde un estado **q** hasta **s**; la expresión **Qi** etiqueta al arco que sale de los **qi**.*
- *Del mismo modo, una expresión regular **Pj** etiqueta el arco desde **s** hasta **pi**, para todo **i**.*
  - *También se ha incluido un bucle sobre **s** con la etiqueta **S**.*
- *Por último, tenemos una expresión regular **Ri j** sobre el arco que va desde **qi** a **pj**, para todo **i** y **j**.*
  - *Observe que algunos de estos arcos pueden no existir en el autómata, en cuyo caso la expresión correspondiente a dicho arco será Ø.*

*La Figura 1.2 muestra lo que ocurre al eliminar el estado **s**. Todos los arcos que implican el estado s se han eliminado. Para compensar esto, para cada predecesor **qi** de **s** y para cada sucesor **pj** de **s** introducimos una expresión regular que represente todos los caminos que comienzan en **qi**, van hasta **s**, quizá hacen un bucle en **s** cero o más veces y, finalmente, llegan a **pj**. La expresión para estos caminos es **QiS*Pj**. Esta expresión se (con el operador unión) al arco que va desde **qi** hasta **pj**. Si no existe ningún arco **qi** → **pj**, entonces añadimos uno con la expresión regularØ.*



***1.1** Un estado s que va a ser eliminado*



1.2** Resultado de eliminar el estado s de la Figura **1.1

III.- Explicar leyes algebraicas de expresiones regulares.

Las leyes algebraicas de las expresiones regulares son propiedades que se cumplen para cualquier expresión regular y que permiten simplificar o transformar las expresiones regulares.

Algunas de las leyes más importantes son:

Asociatividad y Conmutatividad

- Ley conmutativa para la unión: `L+M = M+L`
- Ley asociativa para la unión: `(L+M) + N = L + (M+N)`
- Ley asociativa para la concatenación: `(LM)N = L(MN)`

Identidades y Aniquiladores

- ∅ es la identidad para la unión: ∅ + L = L + ∅ = L
- ε es la identidad para la concatenación: ε L = L ε = L
- ∅ es el aniquilador para la concatenación: ∅L = L∅ = ∅

Leyes Distributivas

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

• Ley distributiva para la concatenación sobre la unión: `L(M+N) = LM + LN`

• Ley Distributiva Derecha para la concatenación sobre unión: **(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 sí aplica como **0 + 0 = 0)**

• En general la multiplicación tampoco es idempotente**: x × x 6= x**

• La unión e intersección son ejemplos comunes de operadores idempotentes. Ley idempotente para la unión: L + L = L

REFERENCIA

http://xamanek.izt.uam.mx/map/cursos/Automatas-HMU08.pdf