Tecnológico de Costa Rica

San José

Compiladores e Interpretes

Apuntes del Viernes 31 de Marzo del 2017

César Borge Mora (2015075361)

Profesor Francisco Torres Rojas

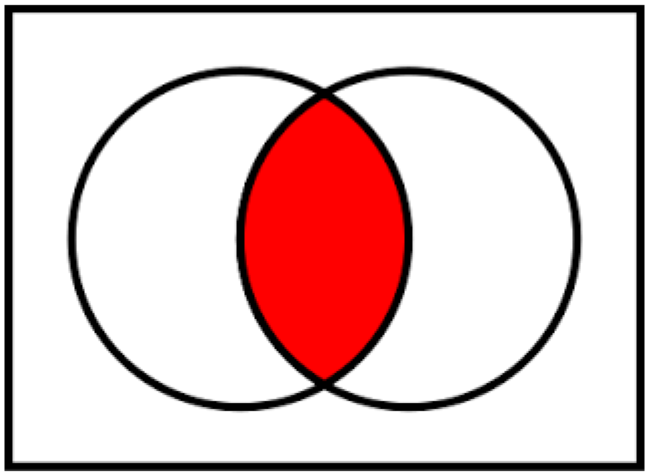
Primer Semestre 2017

Repaso Clase Anterior

Cierre en Intersección de Lenguajes Regulares:

Teorema:

Si L y M son lenguajes regulares entonces $L \cap M$ también es regular.



Los lenguajes regulares son cerrados respecto a la intersección de lenguajes.

Demostración:

$L \cap M = \overline{\overline{L \cap M}}$ (Propiedad del complemento de lenguajes)

$\overline{\overline{L \cap M}} = \overline{\overline{L} \cup \overline{M}}$ (De Morgan)

L y M son lenguajes regulares.

Los lenguajes regulares son cerrados respecto a complemento y respecto a unión.

Por lo tanto los lenguajes regulares son cerrados respecto a intersección.

Nota:

Aquí un video por si alguno desea repasar las Leyes de Morgan:
https://www.youtube.com/watch?v=DZc9SfZdWgg

Augustus De Morgan fue un lógico y matemático británico nacido el 27 de junio de 1806 en Madura (India) y fallecido el 18 de marzo de 1871 en Londres (Inglaterra). Conocido por formular las llamadas Leyes de Morgan, en su memoria, y establecer un concepto riguroso del procedimiento, inducción matemática.

Fin del repaso de la clase anterior.

Cierre en Concatenación de Lenguajes Regulares:

Teorema:

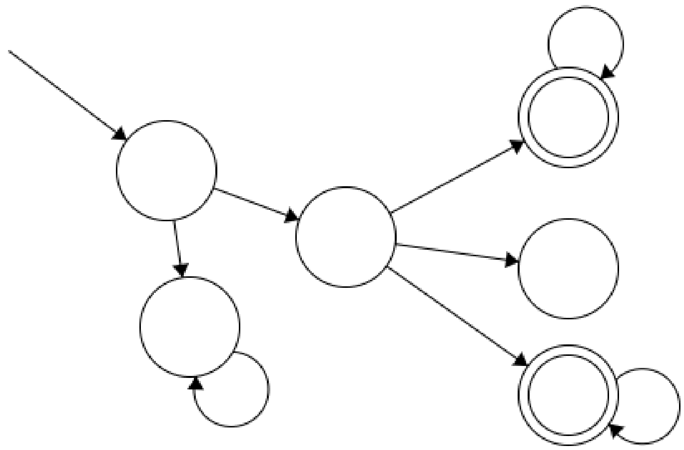
Si L y M son lenguajes regulares entonces LM también es regular.

Los lenguajes regulares son cerrados respecto a la concatenación de lenguajes.

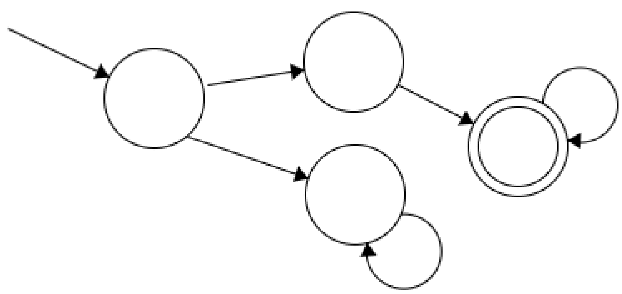
Colochos 1 propuso que para crear un DFA que demostrara esto, se podían unir los estados de aceptación de L con el estado inicial de M. Y que los estados de aceptación del nuevo DFA fueran únicamente los de M.

Y la idea de colochos 1 fue bastante acertada, aquí podemos ver un paso a paso de como hacer un DFA para estos casos utilizando transiciones épsilon:

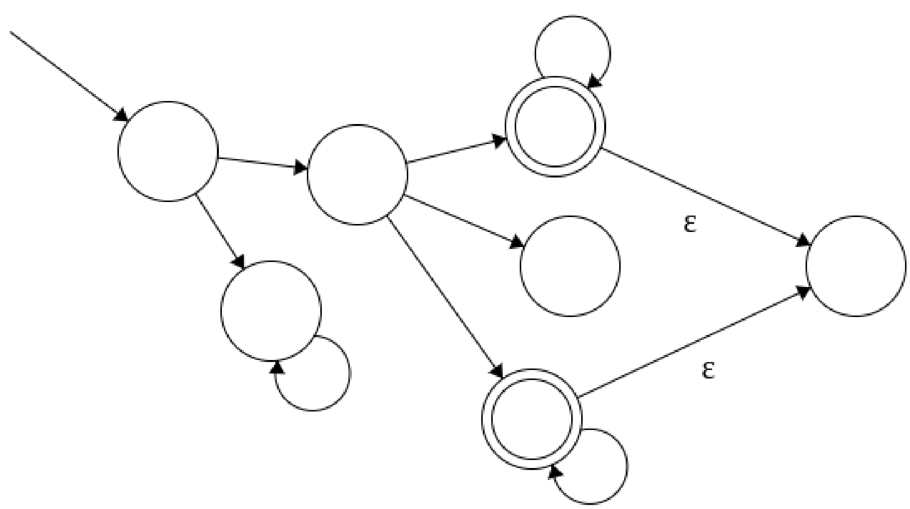
Tenemos este DFA para *L*:



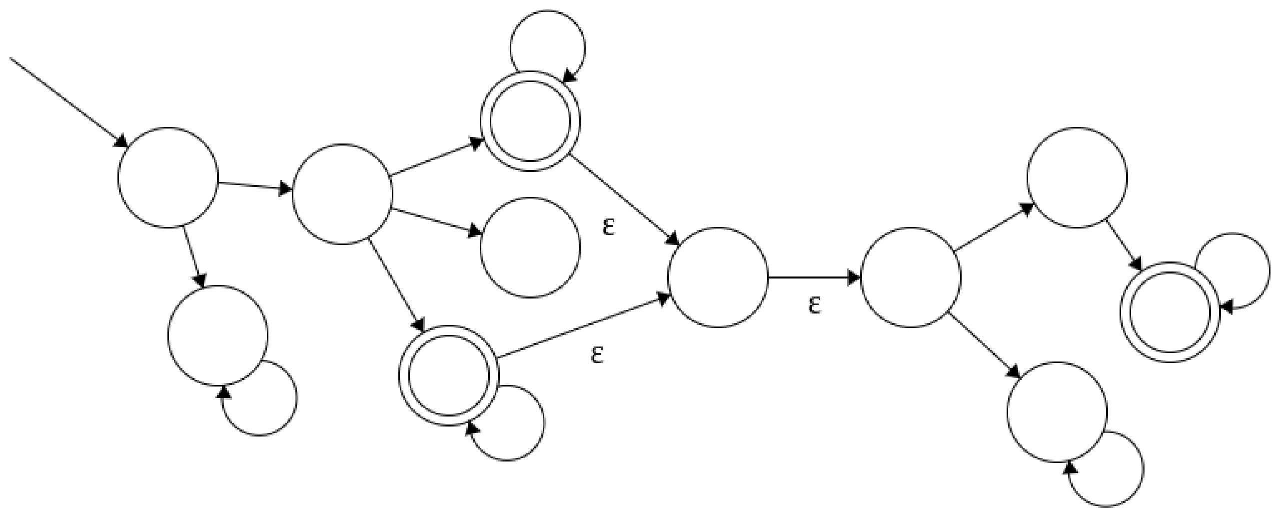
Y este DFA para *M*:



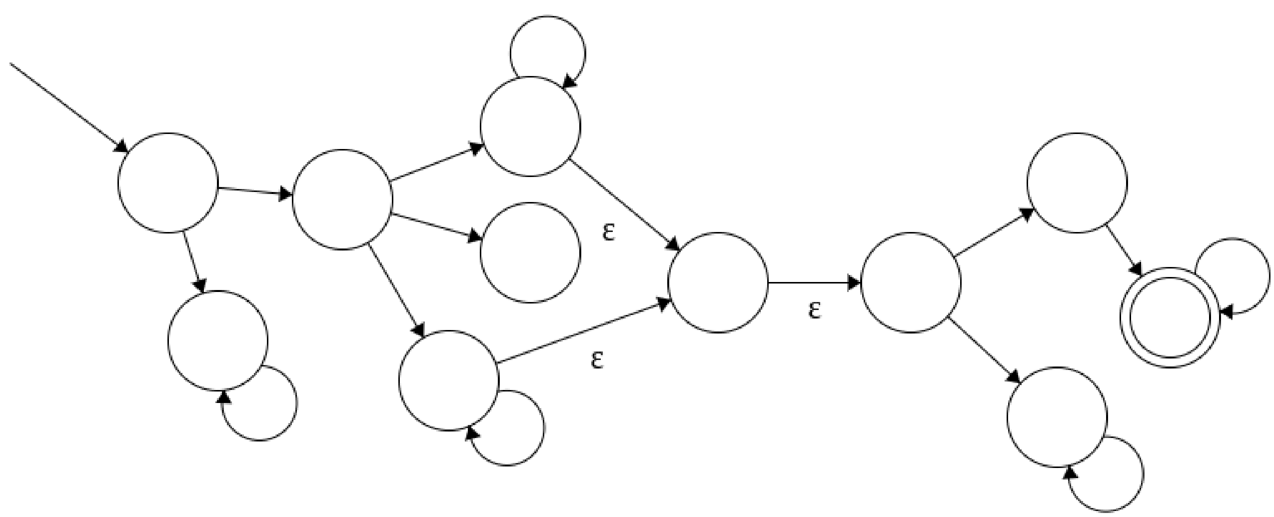
1. Se toman los estados de aceptación de *L* y se unen a un estado nuevo por medio de transiciones épsilon.



2. Se conecta el nuevo estado de *L* con el estado inicial de *M* por medio de una transición épsilon.



3. Los únicos estados de aceptación que se mantienen son los de *M*.

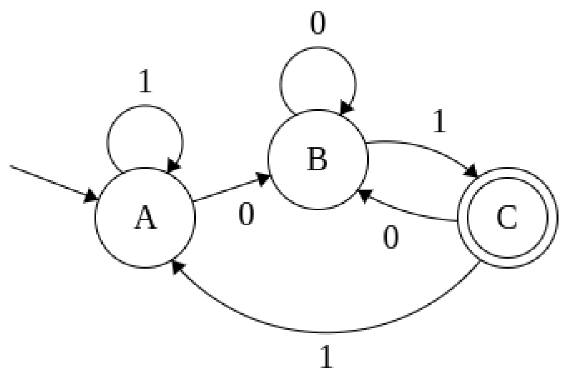


Algo muy interesante sobre esto es que podríamos resolver problemas complejos, dividiéndolos en DFAs más sencillos y luego concatenándolos utilizando esta técnica.

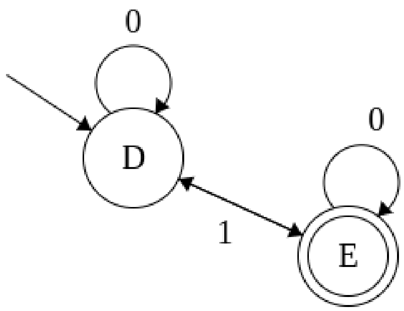
Un ejemplo sería:

Sea L el lenguaje sobre $\sum = \{0,1\}$ de hileras que terminan en 01.
Sea M el lenguaje sobre $\sum = \{0,1\}$ de hileras con un número impar de 1s.

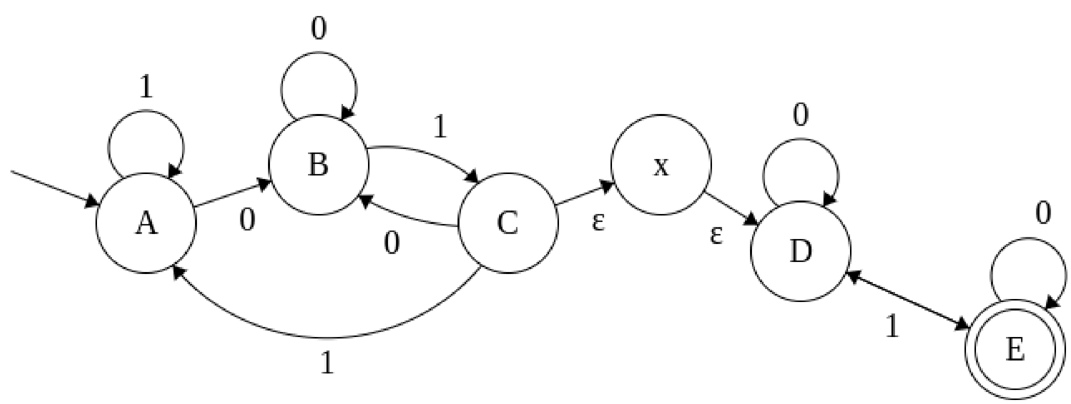
El DFA de L:



El DFA de M:

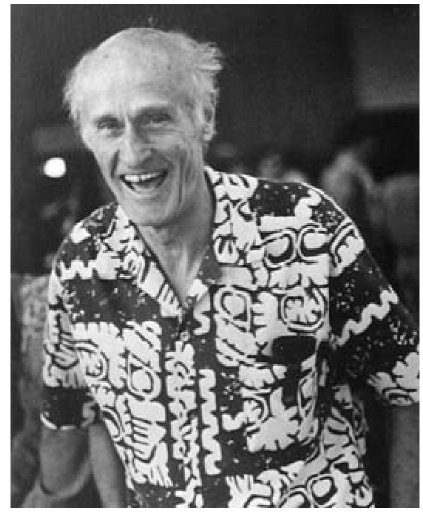


Por lo tanto el DFA de LM es:



Cierre de Kleene de Lenguajes Regulares:

Teorema:



Si L es un lenguaje regular, entonces L^* también es regular.

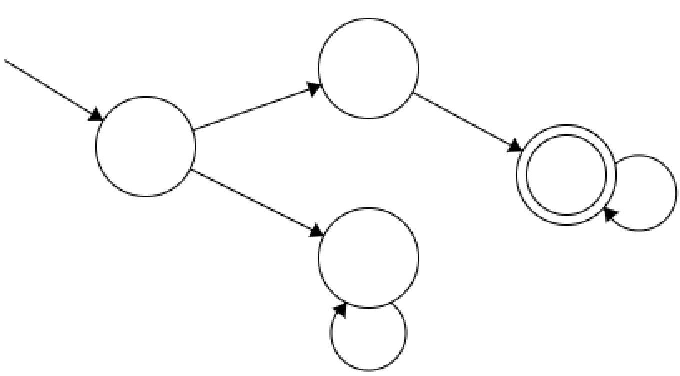
Los lenguajes regulares son cerrados respecto al cierre de Kleene.

Demostración:

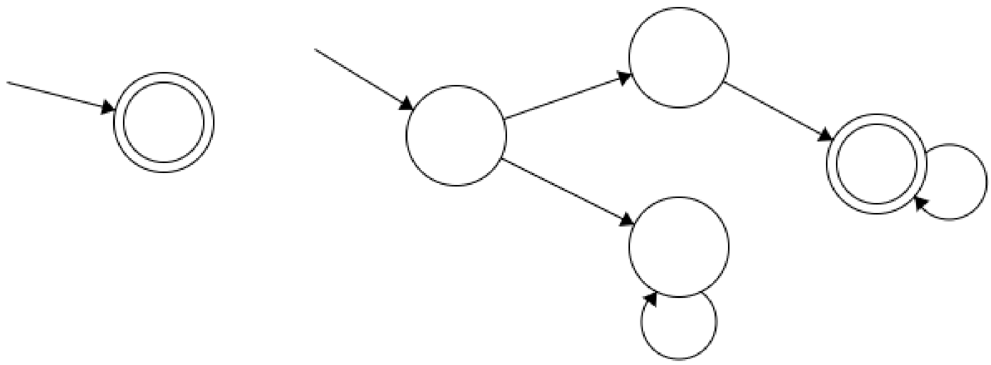
Primero tenemos que tener en cuenta que:

- L es regular.
- L siempre pertenece a L^*.
- Cualquier hilera de L pertenece a L^*.
- Estas hileras se pueden concatenar tantas veces como queramos.
- Si L^* fuera regular, existiría un DFA que lo reconozca.

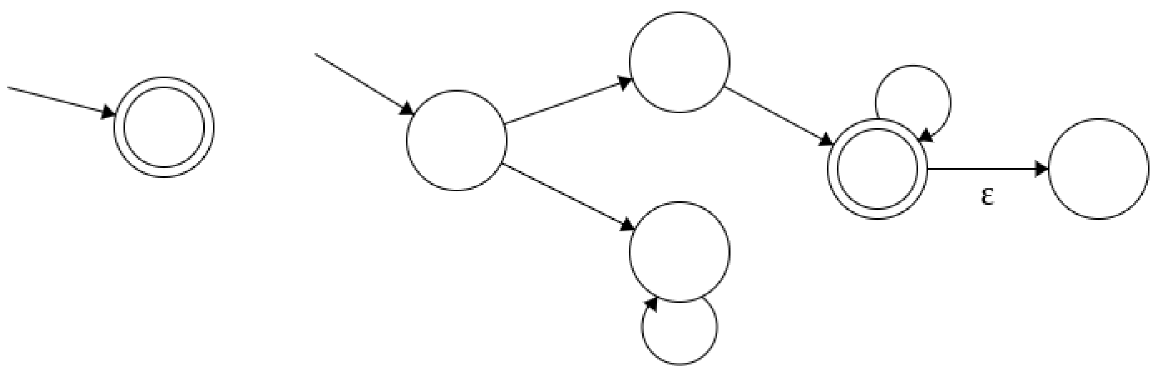
Es importante tener en cuenta que un DFA para L^* siempre va a aceptar épsilon, pues este contiene 0 o más repeticiones del lenguaje L concatenadas. Entonces supongamos que tenemos L:



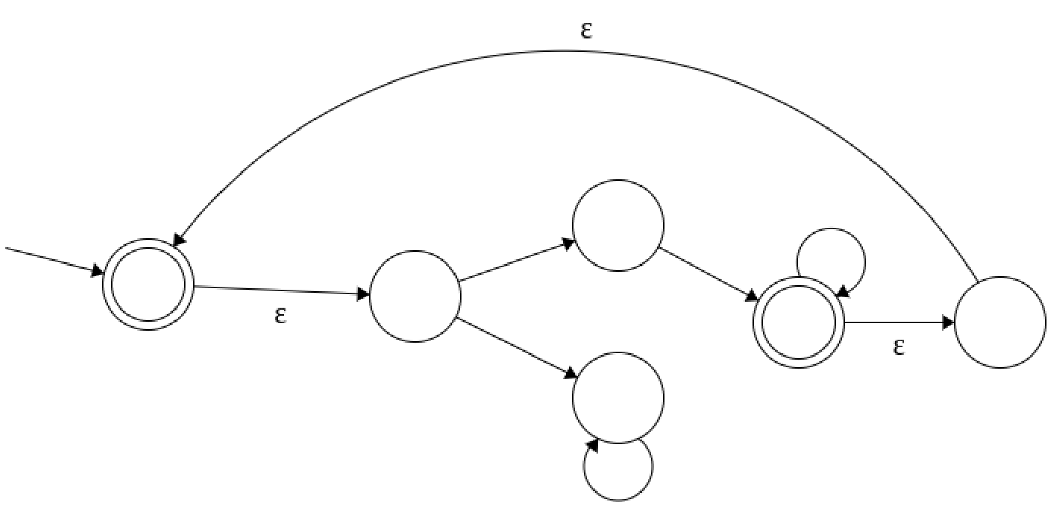
1. Agregamos un nuevo estado inicial que acepte épsilon:



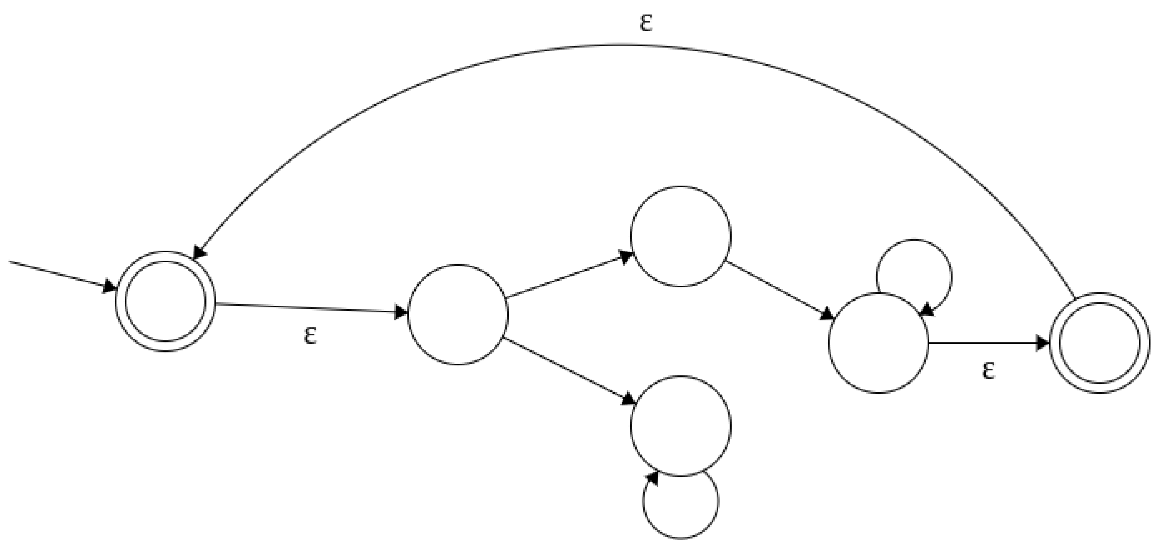
2. Conectamos los estados de aceptación de L a un nuevo estado por medio de una transición epsilon:



3. Conectamos el nuevo estado inicial con el original por medio de una transición épsilon y además tomamos el nuevo estado agregado a L y lo unimos con el nuevo estado inicial por medio de una transición épsilon:



4. Los únicos estados de aceptación serian el nuevo estado inicial y el estado agregado:

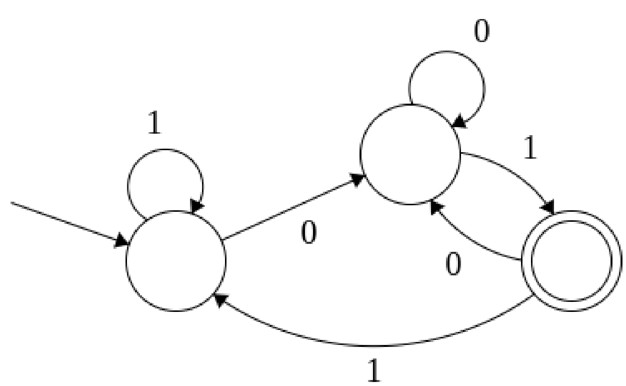


Un ejemplo:

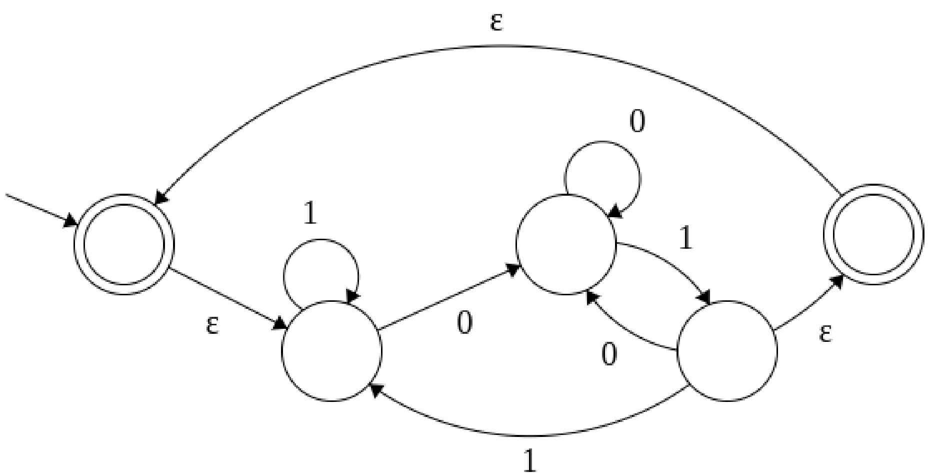
Sea L el lenguaje sobre $\sum = \{0,1\}$ de hileras que terminan en 01.

L^* es regular.

Este es el DFA de L**:**



Este sería el DFA de L^*:



Por lo tanto L^* es regular.

Cierre Cruz de Lenguajes Regulares

Teorema:

Si L es un lenguaje regular, entonces L^+ también es regular.

Los lenguajes regulares son cerrados respecto al cierre cruz.

Torres: ¨Puse esa foto porque es un cierre cruz.¨

Demostración:

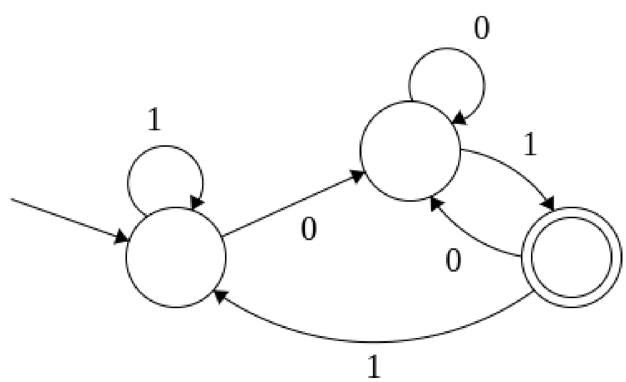
Funciona prácticamente igual a la demostración con L^*, con una diferencia, esta vez no se crea un nuevo estado inicial y el estado nuevo que se agrega, va al estado inicial original.

Entonces un ejemplo:

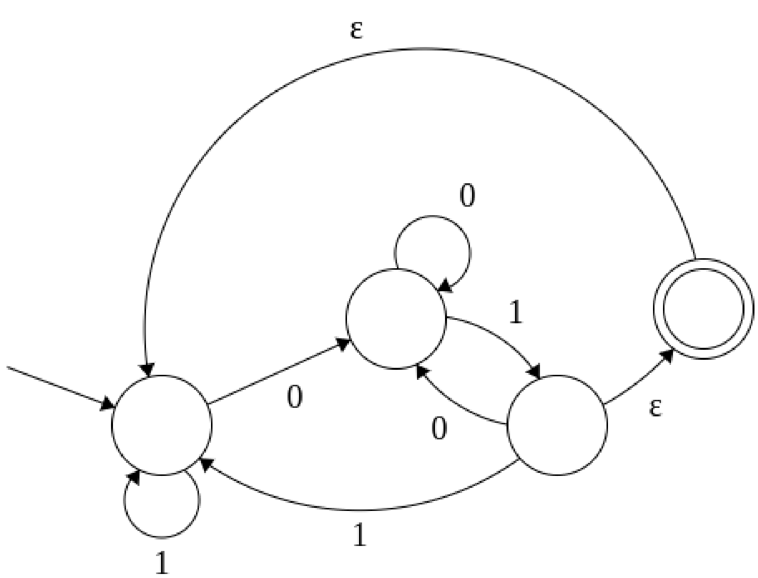
Sea L el lenguaje sobre $\sum = \{0,1\}$ de hileras que terminan en 01.

L^* es regular.

Este es el DFA de L**:**



Así quedaría el DFA de L^+:



Entonces en resumen podemos concluir que los lenguajes regulares son cerrados respecto a:

- Unión
- Complemento
- Intersección
- Concatenación
- Cierre *
- Cierre +

Demostración General:

Podemos diseñar un NFA-ε para cada uno de ellos usando como piezas los DFAs que por definición deben existir para los lenguajes de entrada.

Para demostrar lo contrario se utiliza Pumping Lemma.

Reconocer y Generar

Los lenguajes formales tienen asociados 2 mecanismos:

1. Reconocedores
2. Generadores

Reconocedor:
- Resuelve problemas binarios (acepta o no acepta).
- Acepta todas las hileras del lenguaje.
- Rechaza todas las hileras que no pertenecen al lenguaje.

Generador:
- Puede generar todas las hileras del lenguaje.
- No genera hileras que no pertenezcan al lenguaje.

Entonces los DFAs son **reconocedores** de los lenguajes regulares, pero entonces… ¿Quiénes son los generadores?

Las **expresiones regulares** sirven como **generadores** de los lenguajes regulares.

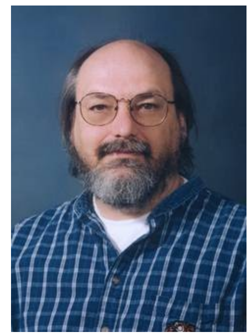
Estas fueron inventadas en 1950 por Stephen Kleene.

Son ampliamente utilizadas en diversos campos de Ciencia de la Computación.

Uso de Expresiones Regulares:

- En 1968, **Ken Thompson** las usó por primera vez en un editor de texto, llamado QED (Quod erat demonstrandum).
- También en este año fueron usadas en los primeros generadores automáticos de analizadores léxicos (Johnson).

UN MOMENTO… es necesario detenerse y hablar más sobre Ken Thompson.



Kenneth Lane Thompson, nació en Nueva Orleans, un 4 de febrero de 1943. Es más conocido como Ken Thompson y se le adjudica junto a **Dennis Ritchie** la creación de **UNIX**.

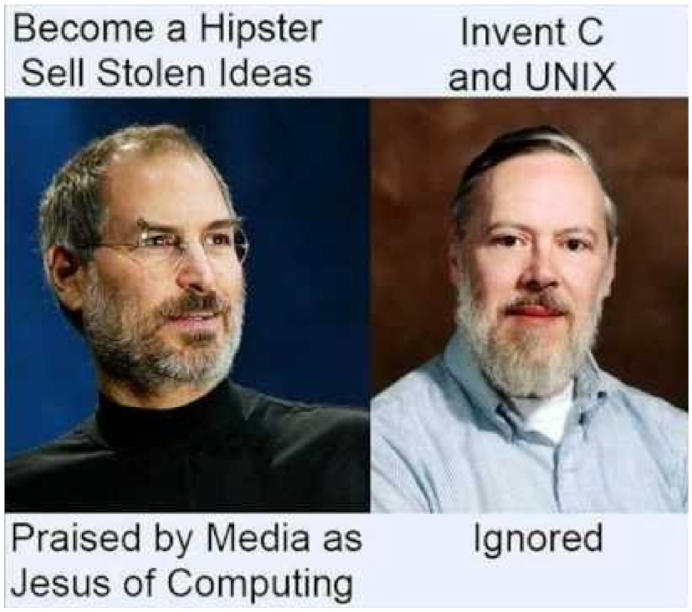
PERO...UN MOMENTO… también tenemos que hablar de Dennis Ritchie.

Dennis MacAlistair Ritchie, nació el 9 de septiembre de 1941 y falleció el 12 de octubre de 2011. Como ya se mencionó anteriormente desarrollo UNIX y también varios lenguajes de programación como **C**. (Si C, ese que usamos tanto, el de segmentation fault)

Ojo que par de mafufos (Just kidding, son unos genios):

Dato Curioso:

Dennis Ritchie falleció unos días después de Steve Jobs (el de Appl...App...perdón casi me rancho), y los medios nunca hablaron de Dennis Ritchie, ni siquiera mencionaron su muerte por estar hablando de Jobs.



Dato todavía más curioso:

¿Sabías que existe un híbrido entre Steve Jobs y Bill Gates, llamado Steve Gates, el cual solo puede ser invocado por el Dr. Francisco Torres?

Se los presento:

Basta de hacerle bullying a Torres!

Teorema de Kleene

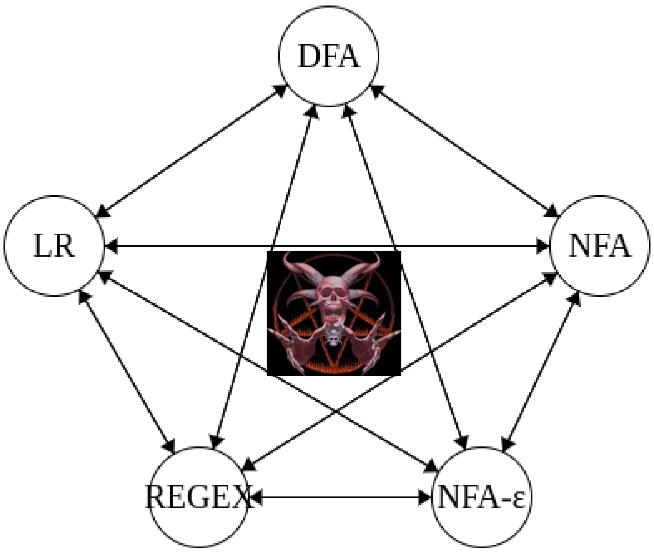
Todo lenguaje regular tiene una expresión regular que lo genera.

Corolario:

Hay una equivalencia entre expresiones regulares y DFAs.

Nota: Uno sabe que un curso es muy volado cuando en todos los apuntes viene esta foto.

Kleene propone una equivalencia entre todo esto:



Expresiones Regulares

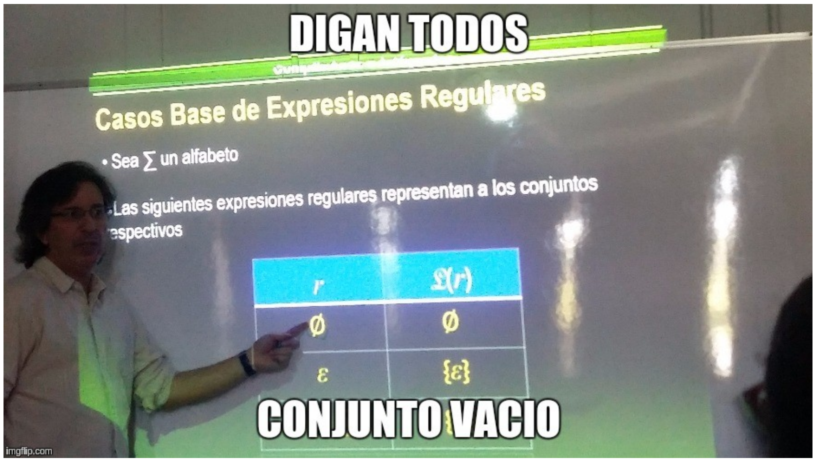
Una expresión regular es un hilera *r* que representa a un lenguaje o conjunto *L(r)*. Osea *r* nos genera *L(r)*.

Casos Base de Expresiones Regulares

Sea ∑ un alfabeto.

Las siguientes expresiones regulares representan a los conjuntos respectivos:

r	$L(r)$
∅	∅
ε	$\{\varepsilon\}$
$a \in \sum$	a



Casos Derivados Expresiones Regulares

Sean r_1 y r_2 expresiones regulares sobre el alfabeto ∑, que representan a $L(r_1)$ y $L(r_2)$.

Las siguientes expresiones regulares representan a los conjuntos respectivos:

r	$L(r)$
$r_1 \cup r_2$	$L(r_1) \cup L(r_2)$
$r_1 r_2$	$L(r_1)L(r_2)$
r_1^*	$L(r_1)^*$

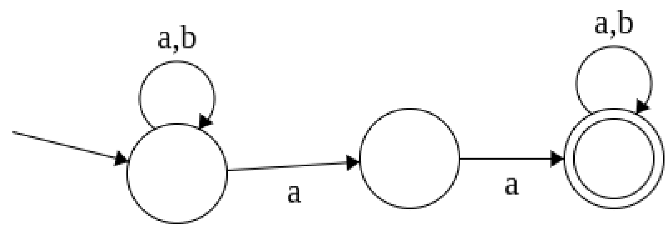
Ejemplo 1:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que contengan la subhilera ¨aa¨

(a ∪ b)*aa(a ∪ *b*)*

o bien

(a | b)*aa(a | *b*)*



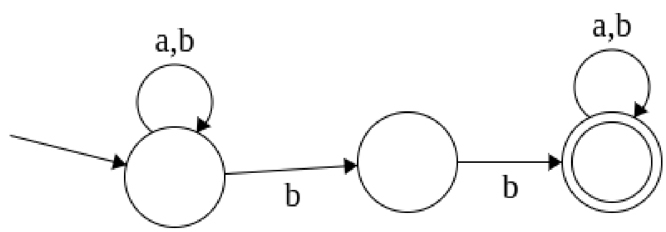
Ejemplo 2:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que contengan la subhilera ¨bb¨

(a ∪ b)*bb(a ∪ *b*)*

o bien

(a | b)*bb(a | *b*)*



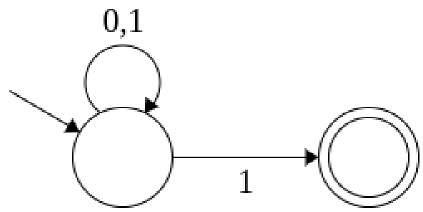
Ejemplo 3:

Expresión regular que genere el lenguaje *L* sobre ∑={0,1} de hileras que terminan en 1

(0 ∪ *1*)*1

o bien

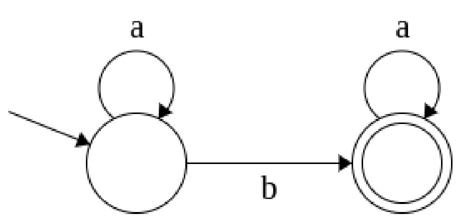
(0 | *1*)*1



Ejemplo 4:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que contengan exactamente una b

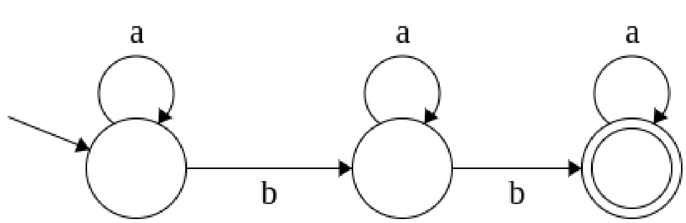
a*ba*



Ejemplo 5:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que contengan exactamente dos b´s

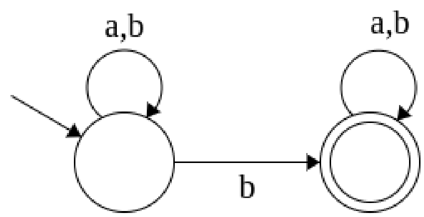
a*ba*ba*



Ejemplo 6:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que contengan al menos una b

(a | b)*b(a|b)*



Ejemplo 7:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que empiecen y terminan con el mismo símbolo

(a(a|b)*a)|(b(a|b)*b)|a|b

Ejemplo 8:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras de longitud par

((a|b)(a|b))*

Ejemplo 9:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b,c} de hileras que no contengan la subhilera bc

c*(b|ac*)*

Ejemplo 10:

Expresión regular que genere el lenguaje *L* sobre ∑={a,b} de hileras que no contengan la subhilera aa

b*(abb*)*|b*(abb*)*a

Ejemplo 11:

Expresión regular que genere el lenguaje *L* sobre ∑={0,1} de hileras que no terminan en 1

(ε|(0|1)*0)

Ejemplo 12:

Expresión regular que genere el lenguaje *L* sobre ∑={0,1} de hileras que terminan en 01

(0|1)*01

Aquí terminan los ejemplos.

Identidades en Expresiones Regulares

- ∅u = u∅ = ∅
- εu = uε = u
- ∅* = ε
- ε* = ε
- u|v = v|u
- a|a = a
- u* = (u*)*
- u(v|w) = uv | uw
- (u|v)w = uw|vw
- (uv)*u = u(vu)*
- (u|v)* = (u*|v)* = u*(u|v)* = (u|vu*)* = (u*v*)* = u*(vu*)* = (u*v)*u*

Peter Gustav Lejeune Dirichlet

Fue un matemático alemán que nació en 1805 y murió en 1859. Es considerado uno de los matemáticos más importantes de la historia. Se le adjudica la notación actual de función. Trabajó en la Teoría de Números, Series de Fourier, Análisis Matemático y en el **Principio del Palomar.**

Principio del Palomar (Principio de Dirichlet):

Si tengo un palomar con 9 casetillas para palomas y colocó 10 palomas, necesariamente alguna casetilla tendrá más de una paloma.

Otra manera de decirlo es:

Si tengo que acomodar n objetos en m recipientes, con $n > m$, entonces necesariamente alguna recipiente tendrá más de un objeto.

Y la manera formal de decirlo es:

No existe ninguna funcion inyectiva cuyo codominio sea más pequeño que su dominio.

Este principio por simple que parezca, tiene resultados muy importantes en matemáticas, computación, biología, economía, compactación de datos, criptografía, comunicaciones, etc.

Veamos varios ejemplos:

1. Cabellos en la Cabeza

Se estima que una cabeza humana tiene a lo más 500 mil cabellos. Si la población de Costa Rica es de 5 millones, ¿Cuál es la probabilidad de que en Costa Rica haya 2 personas con exactamente la misma cantidad de cabellos en sus cabezas?

Los pelones también cuentan

La respuesta es 1.0, es totalmente cierto. Esto porque hay 500000 categorías en las que se podría colocar a las personas según la cantidad de pelos en su cabeza. Hay 5 millones de personas y por ende en alguna categoría va a quedar más de una persona.

2. Iniciales en el TEC

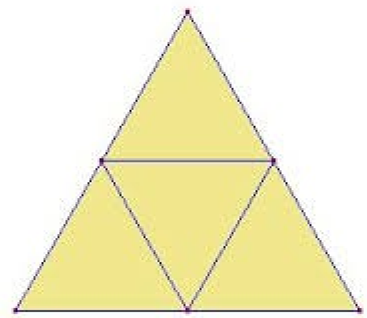
¿Habrá 2 estudiantes de Computación del TEC que tengan las mismas iniciales? ¿En el TEC? ¿En Costa Rica?

Si el alfabeto tiene 27 letras, y las iniciales de un nombre como máximo van a tener 4 letras. 27 * 27 * 27 * 27 = 531441.

Es probable que entre los estudiantes de Computación y el TEC en total esto no se cumpla, pues hay más categorías de iniciales (531441) que estudiantes en ambos casos. Sin embargo tomando la totalidad de la población de Costa Rica esto se va a cumplir con una probabilidad de 1.0 .

Tiro al Triángulo

En la pared hay un triángulo equilátero de 40cm de lado hecho de corcho. Si tiramos y clavamos 5 dardos en este triángulo, al menos 2 estarán a una distancia menor o igual a 20cm.



Esto se da porque si partimos el triángulo en triángulos iguales, nos van a quedar 4 triángulos para colocar 5 dardos. Cada área de cada triangula tiene una distancia máxima de 20cms. Otra vez por Principio del Palomar esto tiene una probabilidad de 1.0.

Suma 11

Dados 6 números diferentes entre 1 y 10, siempre hay 2 que suman 11.

Esto es cierto por el Principio del Palomar ya que entre los números 1 y 10 hay 5 parejas de números que suman 11:

- 1 y 10
- 2 y 9
- 3 y 8
- 4 y 7
- 5 y 6

Al escoger 6 números diferentes al menos 2 son de una misma pareja de las anteriores.

Finalmente recuerden:

- Tenemos entrega de proyecto este miércoles.
- Tenemos examen este viernes y el profe va a permitir que saquemos un forro de una página tamaño carta, escrito a mano por un solo lado.
- Tarea de expresiones regulares para el viernes.