Universidad Politécnica de Aguascalientes

**Materia:** Lenguajes y Autómatas.

**Tarea:** Ensayo Unidad 3.

**Profesor:** Dr. Christian José Correa Villalón.

**Alumno:** Juan Carlos Pedroza Hernández.

**Matrícula:** UP170132.

**Grupo:** ISEI007.

**Fecha de Entrega:** 09/04/2020

Contenido

[Introducción: 2](#_Toc37264949)

[Lenguajes regulares sobre un alfabeto 3](#_Toc37264950)

[Propiedades de los lenguajes regulares 4](#_Toc37264951)

[Lema de Pumping 4](#_Toc37264952)

[Propiedades de Cerradura 6](#_Toc37264953)

[Equivalencia y minimización de autómatas 6](#_Toc37264954)

[Expresiones regulares 7](#_Toc37264955)

[Operaciones de los lenguajes: 7](#_Toc37264956)

[Operandos 8](#_Toc37264957)

[Precedencia: 8](#_Toc37264958)

[Identidades y Aniquiladores 11](#_Toc37264959)

[Leyes distributivas 11](#_Toc37264960)

[Conclusión: 12](#_Toc37264961)

[Bibliografía 12](#_Toc37264962)

# Introducción:

Con este ensayo abarcaremos toda la unidad 3 hablando sobre los temas que debemos abarcar en la unidad.

Se hará una investigación de los distintos temas, lenguajes regulares, expresiones regulares, equivalencia de expresiones regulares y gramáticas regulares, con está investigación podremos tener conocimiento de cada uno de los temas.

Lenguajes Regulares y Expresiones Regulares

Son los que se pueden generar partir de los lenguajes básicos, con la aplicación de operaciones de unión, concatenación y \* de Kleene un número finito de veces.

Pueden ser reconocidos por:

* Un autónomo finito determinista.
* Un autómata finito no determinista.
* Un autómata de pila.
* Un autómata finito alterno
* Una máquina de Turing de solo lectura.

# Lenguajes regulares sobre un alfabeto

Un lenguaje regular sobre un alfabeto ∑ dado se define recursivamente como:

* El lenguaje regular Ø es un lenguaje regular.
* El lenguaje cadena vacía {} es un lenguaje regular.
* Para todos los símbolos a € ∑ es un lenguaje regular.
* Si A y B son lenguajes regulares A U B (unión), A \* B (concatenación) y A\* (clausura o estrella de Kleene) son lenguajes regulares.
* Si A es un lenguaje regular entonces (A) es el mismo lenguaje regular.

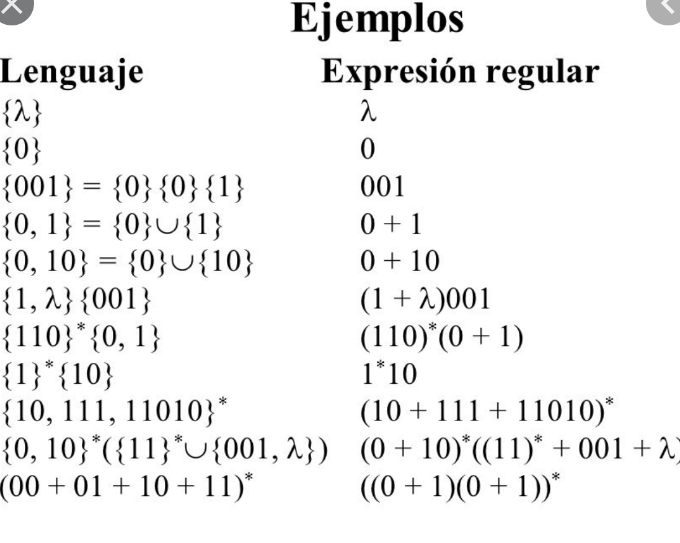
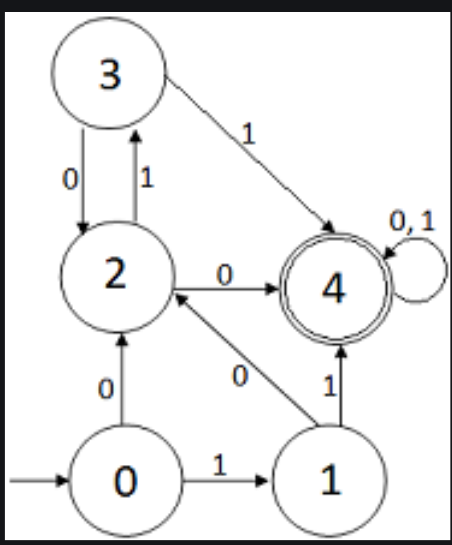
 

Ilustración : Expresiones regulares. Ilustración : Autómata finito.

# Propiedades de los lenguajes regulares

Existen distintas herramientas que se pueden utilizar sobre los lenguajes regulares:

* El **Lema Pumping**: cualquier lenguaje regular satisface el Lema de Pumping, el cual se puede usar para probar que un lenguaje no es regular.
* **Propiedades de cerradura**: se pueden construir autómatas a partir de componentes usando operaciones, por ejemplo, dado un lenguaje L y M construir un autómata para L ∩ M.
* **Propiedades de decisión:** análisis computacional de autómatas. Por ejemplo, probar si dos autómatas son equivalentes.
* **Técnicas de minimización:** útiles para construir máquinas más pequeñas.

La clase de lenguajes conocidos como *lenguajes regulares* tienen al menos 4 descripciones: DFA, NFA, €-NFA y RE.

No todos los lenguajes son regulares, por ejemplo: L = {0 n1 n |n ≥ 1}.

# Lema de Pumping

Si es un lenguaje regular, entonces existe una constante tal que cada cadena , de longitud o más, puede ser escrita como , donde:

1. .
3. Para toda , también está en . Nótese que repetida veces; .

Lo que dice este lema es que, si tenemos una cadena con una longitud mayor al número de estados del autómata, entonces una cadena no vacía puede ser repetida (“pumped”) un número arbitrario de veces.

Algunas consideraciones importantes son:

* Como se da por hecho que L es regular, debe existir un DFA A tal que L=L(A). Si A tiene n estados; escogemos esta n para el Lema de Pumping.
* Sea w una cadena de longitud ≥ n en L, por ejemplo, w = a1a2 . . . am, donde m ≥ n.
* Sea qi el estado en que A esta después de leer los primeros i símbolos de w.
* q0 = estado de inicio, q1 = (q0, a1), q2 = (q0, a1a2), etc.
* Como solo hay n estados diferentes, dos de q0, q1, . . ., qn deben ser los mismos; digamos qi = qj, donde 0 ≤ i < j ≤ n.
* Sea x = a1 . . . ai; y = ai+1 . . . aj; z = aj+1 . . . am. Entonces, si repetimos el ciclo desde qi a qi con la etiqueta ai+1 . . . aj cero o más veces, se puede probar que xyiz es aceptado por A.

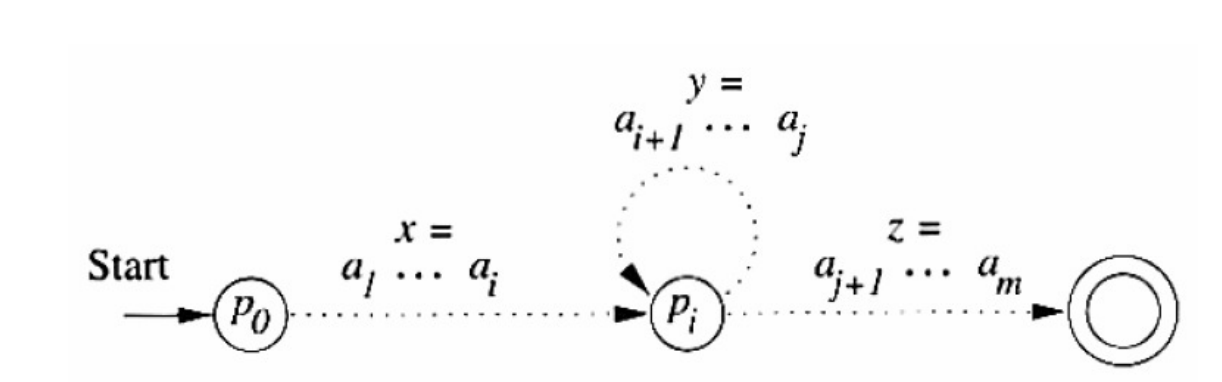


Ilustración : Lema de Pumping.

El uso de Pumping se utiliza para mostrar que un lenguaje L no es regular.

* Se inicia suponiendo que L es regular.
* Luego, debe haber alguna n que sirve como constante de PL (puede que no sepamos el valor n).
* Escogemos una w que sabemos que está en L (normalmente w depende de n).
* Aplicando el PL, sabemos que w puede descomponerse en la forma xyz, satisfaciendo las propiedades del PL (de nuevo, puede que no sepamos cómo descomponer w, así que utilizamos x, y, z como parámetros).
* Derivamos una contradicción escogiendo i (la cual puede depender de n, x, y, y/ o z) tal que xyiz no está en L.

**Ejemplo:**

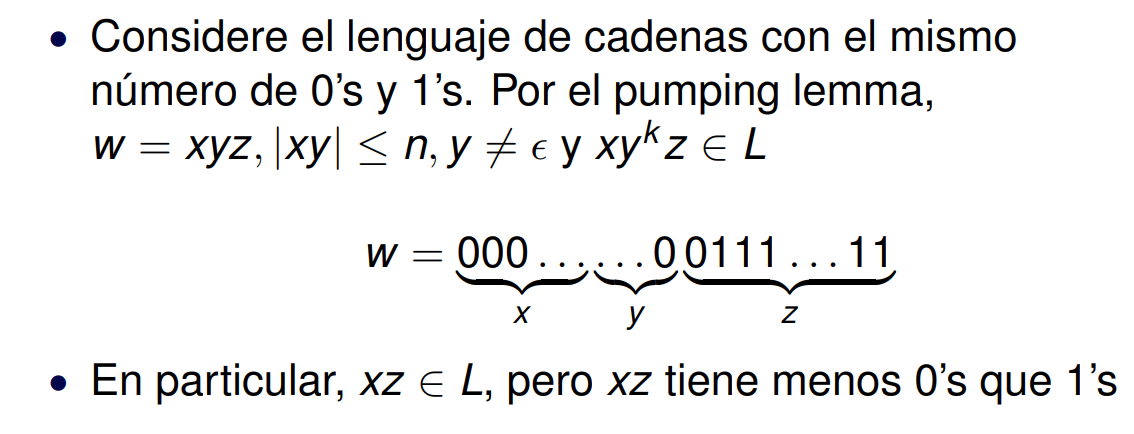


Ilustración : Ejemplo 1.

# Propiedades de Cerradura

**Unión:** la unión de lenguajes regulares es regular. Sea L=L€ y M = L(F). Entonces L (E + F) = L U M, pro la definición de “+” en RE.

**Complemento:** Si L es un lenguaje regular sobre ∑, entonces también lo es L’ = ∑ \* L. Todos los estados son de aceptación excepto los F.

**Ejemplo:**

Sea L definido por el siguiente DFA (el lenguaje de cadenas que terminan en 01):

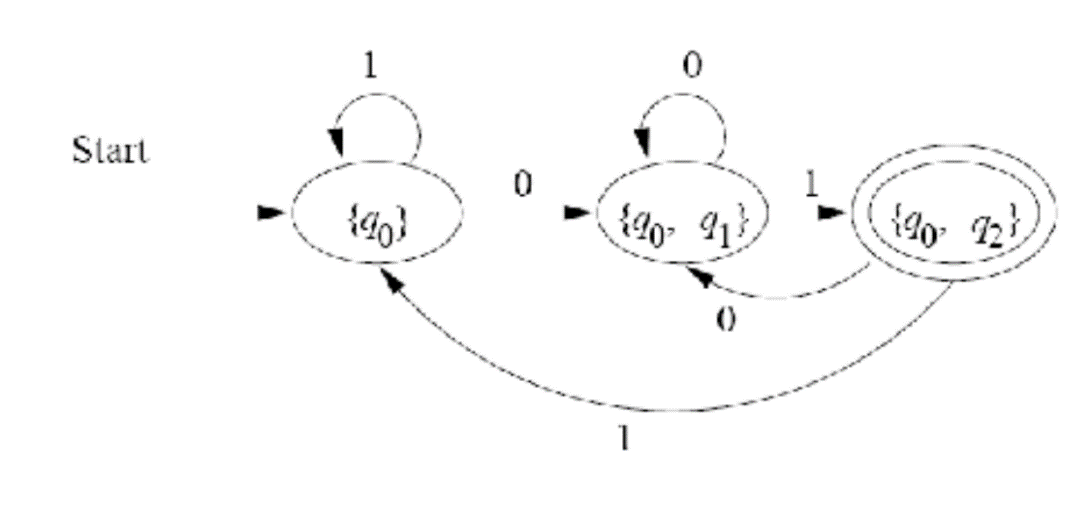


Ilustración : Ejemplo 2 propiedades de cerradura.

# Equivalencia y minimización de autómatas

* Lo que queremos saber es si dos autómatas diferentes definen el mismo lenguaje.
* Primero definiremos lo que son estados **equivalentes**:
* Dos estados p y q dentro de un autómata son equivalentes si: p ≡ q ⇔ ∀w ∈ Σ ∗: δˆ (p, w) ∈ F ⇔ δˆ (q, w) ∈ F.
* Si no, entonces se dice que son **distinguibles**. Es decir, p y q son distinguibles si:

∃w: δˆ (p, w) ∈ F ∧ δˆ (q, w) ∈/ F o viceversa.

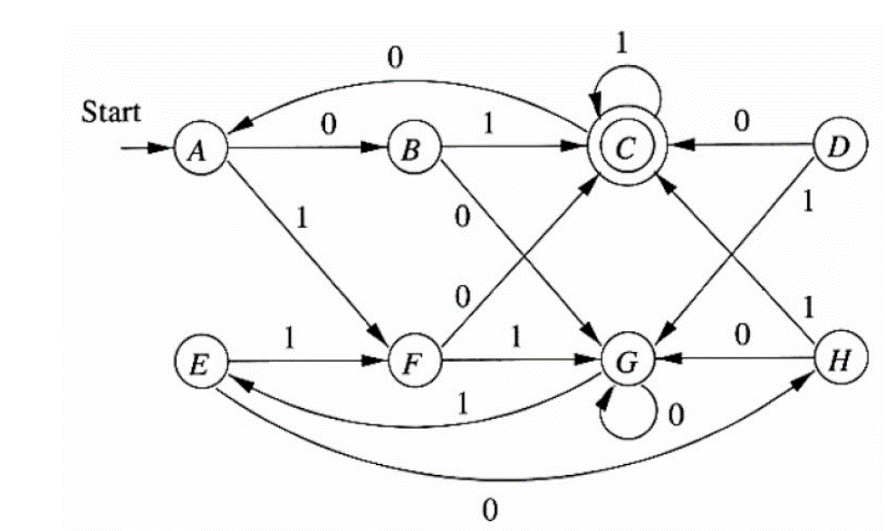


Ilustración : Ejemplo de propiedades de cerradura.

# Expresiones regulares

* Es un equivalente algebraico para un autómata.
* Utilizado en muchos lugares como un lenguaje para describir patrones en texto que son sencillos pero muy útiles.
* Pueden definir exactamente los mismos lenguajes que los autómatas pueden describir: lenguajes regulares.
* Ofrecen algo que los autómatas no: Maneras declarativas de expresar las cadenas que queremos aceptar.

**Ejemplos de usos:**

* Comandos de búsqueda, por ejemplo, grep de UNIX.
* Sistemas de formateo de texto: usan notación de tipo expresión regular para describir patrones.
* Convierte la expresión regular a un DFA o un NFA y simula el autómata en el archivo de búsqueda.
* Generadores de analizadores-léxicos, como Lex o Flex.

Las expresiones regulares denotan lenguajes. Por ejemplo, las expresiones regulares: 01\*+ 10\* denotan todas las cadenas que no son o un 0 seguido de cualquier cantidad de 1’s o un 1 seguido de cualquier cantidad de 0’s.

# Operaciones de los lenguajes:

1. Unión.
2. Concatenación
3. Cerradura (o cerradura de Kleene)

Si E es una expresión regular, entonces L(E) denota el lenguaje que define E. Las expresiones se construyen de la manera siguiente:

* Las constantes € y ∅ son expresiones regulares que representan al lenguaje L (€) = {€} y L (∅) = ∅ respectivamente.
* Si a es un símbolo, entonces es una expresión regular que representa al lenguaje: L(a) = {a}.

# Operandos

1. Si E y F son expresiones regulares, entonces E + F también lo es denotando la unión de L(E) y L(F) como L (E + F) = L(E) ∪ L(F).
2. Si E y F son expresiones regulares, entonces EF también lo es denotando la concatenación de L(E) y L(F) como L(EF) = L(E)L(F).
3. Si E es una expresión regular, entonces E∗ también lo es y denota la cerradura de L(E) es decir L (E ∗) = (L(E)) ∗
4. Si E es una expresión regular, entonces (E) también lo es. Formalmente tenemos L((E)) = L(E).

# Precedencia:

1. El asterisco de la cerradura tiene la mayor precedencia
2. Concatenación sigue en precedencia a la cerradura, el operador “dot”. Concatenación es asociativa y se ´ sugiere agrupar desde la izquierda (i.e. 012 se agrupa (01)2).
3. La unión (operador ´ +) tiene la siguiente precedencia, también es asociativa.
4. Los paréntesis pueden ser utilizados para alterar el agrupamiento.

**Ejemplos:**

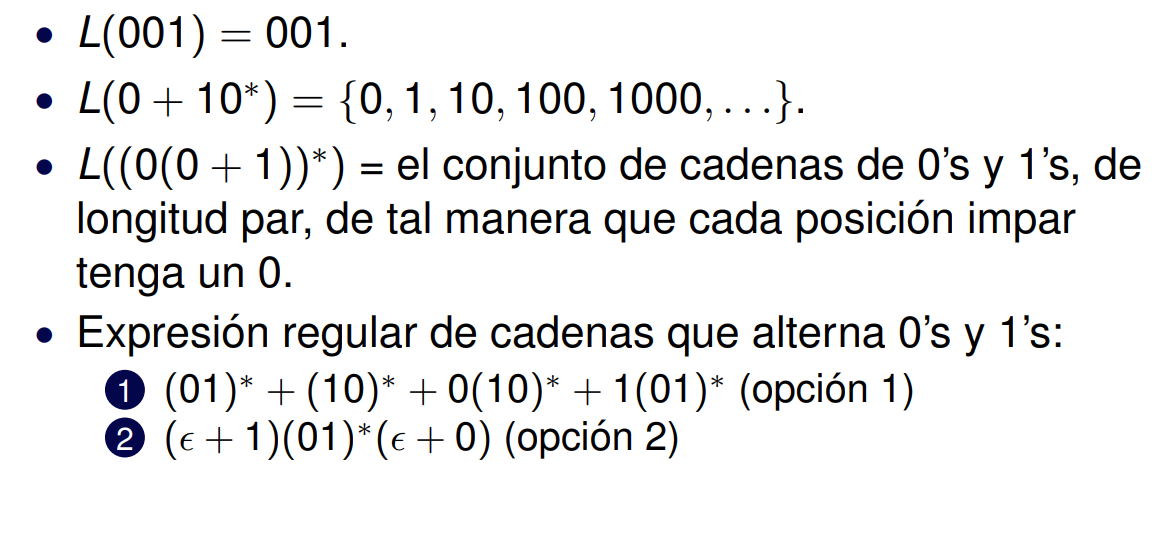


Ilustración : Ejemplos de expresiones regulares.

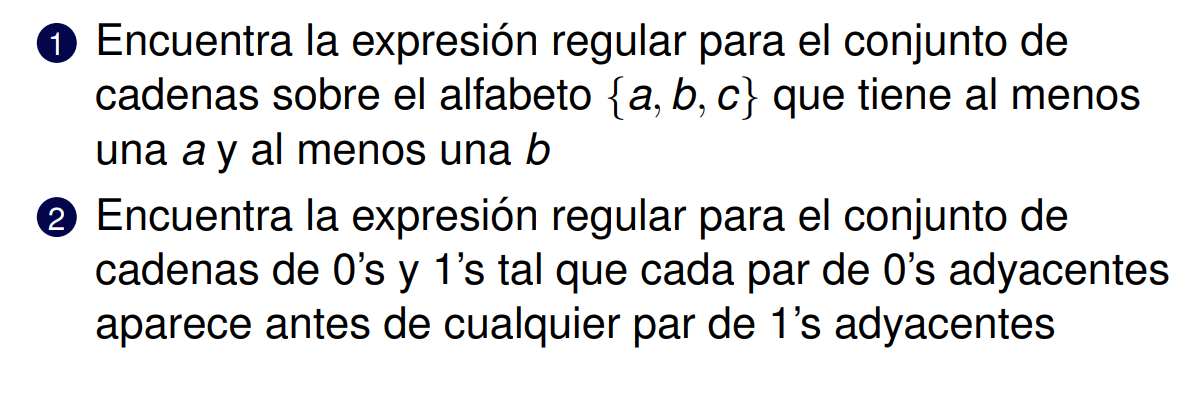


Ilustración : Ejemplos 1 y 2.

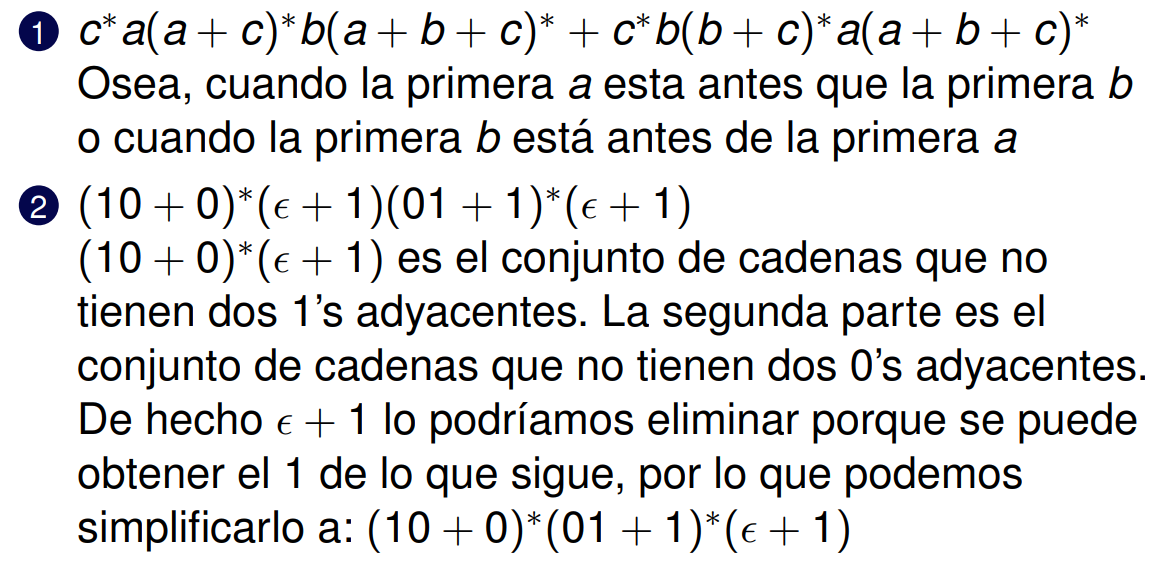


Ilustración : Resultado 1 y 2.

Equivalencia de Expresiones Regulares

DFA’s

* Si l=L(A) para algún DFA A, entonces existe una expresión regular suponiendo que A tiene estados {1, 2, . . ., n}, n finito. Tratemos de construir una colección de RE que describan progresivamente conjuntos de rutas del diagrama de transiciones de A.
* R (k)ij es el nombre de la RE cuyo lenguaje es el conjunto de cadenas w.
* w es la etiqueta de la ruta del estado i al estado j de A. Esta ruta no tiene estado intermedio mayor a k. Los estados inicial y terminal no son intermedios, i y/o j pueden ser igual o menores que k.
* Para construir R(k)ij se utiliza una definición inductiva de k = 0 hasta k = n.
* Base: k = 0, implica que no hay estados intermedios. Solo dos clases de rutas cumplen con esta condición:

1. Un arco del nodo (estado) i al nodo j.
2. Una ruta de longitud 0 con un solo nodo i.

* Si i 6= j, solo el caso 1 es posible.

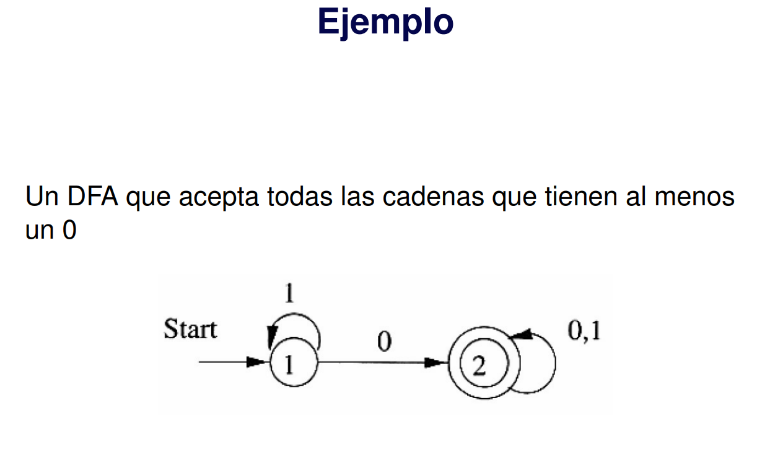


Ilustración Ejemplo DFA.

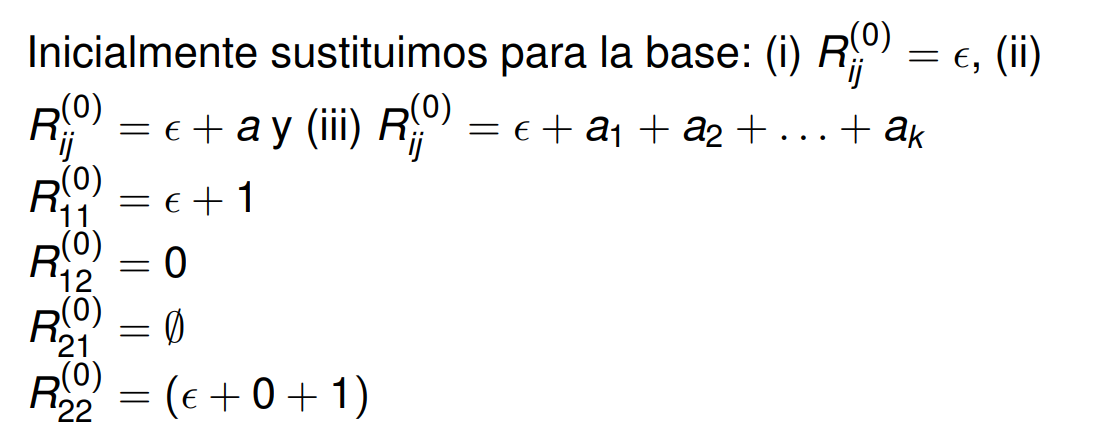


Ilustración : Solución.

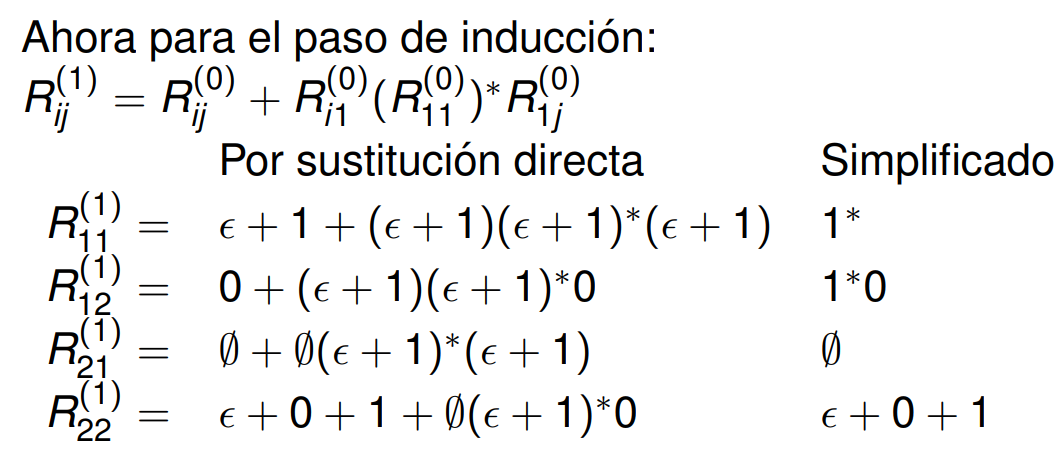


Ilustración : Solución.

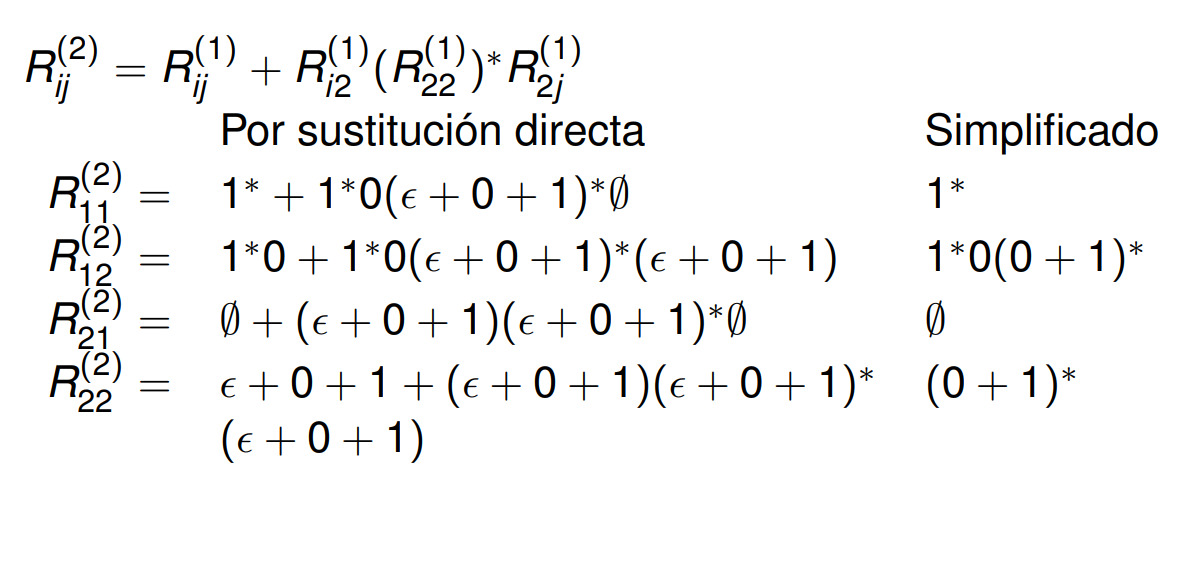


Ilustración : Solución.

Gramáticas Regulares

Existen un conjunto de leyes algebraicas que se pueden utilizar para las expresiones regulares:

* 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

* Una identidad para un operador es un valor tal que cuando el operador se aplica a la identidad y a algún´ otro valor, el resultado es el otro valor.
* 0 es la identidad para la adición: 0 ´ + x = x + 0 = x.
* 1 es la identidad para la multiplicación: ´ 1 × x = x × 1 = x
* Un aniquilador para un operador es un valor tal que cuando el operador se aplica al aniquilador y algún otro valor, el resultado es el aniquilador.
* 0 es el aniquilador para la multiplicación: ´ 0 × x = x × 0 = 0
* No hay aniquilador para

# Leyes distributivas

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

* **Ley Distributiva Izquierda** para la concatenación sobre unión:

L (M + N) = LM + LN.

* **Ley Distributiva Derecha** para la concatenación sobre unión:

(M + N) L = ML + N.

# Conclusión:

Con esta investigación pude obtener el conocimiento para poder abordar estos temas y poder tener noción de lo que se está hablando.

# Bibliografía

*http://decsai.ugr.es/~rosa/tutormc/teoria/EXPRESIONES%20REGULARES.htm*. (s.f.). Obtenido de http://decsai.ugr.es/~rosa/tutormc/teoria/EXPRESIONES%20REGULARES.htm.

*https://ccc.inaoep.mx/~emorales/Cursos/Automatas/ExpRegulares.pdf*. (s.f.). Obtenido de https://ccc.inaoep.mx/~emorales/Cursos/Automatas/ExpRegulares.pdf.

*https://ccc.inaoep.mx/~emorales/Cursos/Automatas/PropsLengRegulares.pdf*. (s.f.). Obtenido de https://ccc.inaoep.mx/~emorales/Cursos/Automatas/PropsLengRegulares.pdf.

*https://prezi.com/l53byacorq8h/lenguajes-regulares-expresiones-regulares-y-gramatica-regul/*. (s.f.). Obtenido de https://prezi.com/l53byacorq8h/lenguajes-regulares-expresiones-regulares-y-gramatica-regul/.