

# LENGUAJES Y AUTÓMATAS I

SCO - 1015

2-3-5

**Comportamiento a desarrollar:** Define, diseña y programa las fases del analizador léxico y sintático de un traductor o compilador, como preámbulos de la construcción de un compilador.

## UNIDAD 1: INTRODUCCIÓN A LA TEORÍA DE LENGUAJES FORMALES.

1.1 Alfabeto

1.2 Cadenas

1.3 Lenguajes, tipos y herramientas

1.4 Estructura de un traductor

1.5 Fases de un compilador

## UNIDAD 2: EXPRESIONES REGULARES (ER)

2.1 Definición formal de una ER

2.2 Diseño de ER

2.3 Aplicaciones en problemas reales

## UNIDAD 3: AUTÓMATAS FINITOS (AF)

3.1 Conceptos: Definición y clasificación de automatas F.

3.2 Conversión de un Autómata F. no determinista (AFN) a un Autómata Finito Determinista (AFD)

3.3 Representación de ER usando AFN

3.4 Minimización de estados AFD

3.5 Aplicaciones

## UNIDAD 4: ANÁLISIS LÉXICO

4.1 Funciones del analizador léxico

4.2 Componentes del analizador léxico

4.3 Creación de tabla de tokens

4.4 Errores léxicos

4.5 Generadores de analizadores léxicos

4.6 Aplicaciones

## UNIDAD 5: ANÁLISIS SINTÁTICO

5.1 Definición y clasificación de gramáticas

5.2 Gramáticas libres de contexto (GLC)

5.3 Árboles de derivación

5.4 Forma normal de Chomsky

5.5 Diagramas de sintaxis

5.6 Eliminación de la ambigüedad

5.7 Tipos de analizadores

5.8 Manejos de errores

5.9 Generadores de analizadores sintáticos

## UNIDAD 6: MÁQUINAS DE TURING (MT)

6.1 Definición formal de una máquina de Turing

6.2 Construcción modular de una máquina de Turing

6.3 Lenguajes aceptados con la MT

## BIBLIOGRAFIA (SUGERIDA EN INTERNET)

• Teoría de Automatos y lenguajes formales

• Compiladores

• Teoría de la computación

• Lenguajes formales

• Compiladores: principios técnicos y herramientas

(biblioteca) (Pearson educación) (Aho Alfred)

Introducción a la teoría de automátos lenguajes

y computación (Madrid, Addison Wesley)

Hopcroft, John

- Teoría de los automatos y lenguajes formales (Prentice Hall, Kelley, D) 1995  
 • Lenguajes formales y teoría de la computación (Mc Graw Hill, Martín J) 2004
- Forma de evaluar:

**UNIDAD 1:**

- Examen diagnóstico 10%
- Act. clase y tareas 20%
- Exposición defectos del compilador 30%
- Examen 40%

**UNIDAD 2 y 3:**

- Act. clase y tareas 30%
- Manual de Proc. expresiones regulares 30%
- Examen 40%

**UNIDAD 4 y 5:**

- Manual de analizador léxico 30%
- Exposición 20%
- Proyecto final 50%

**UNIDAD 5:**

- M. del analizador sintático 30%
- Exposición 30%
- Proyecto final 40%

**UNIDAD 6:**

- Automatización de la configuración de la cinta de una máquina Turing 30%
- Exposición del ejercicio 30%
- Examen 40%

Fecha de exámenes: 14 al 18 de octubre  
19 al 25 de noviembre  
13 al 17 de enero

Exámenes de recuperación: 20 al 22 de enero

- ①) REGLAS DE CLASE:
- \* Asistencia - firmas no se recuperan
  - \* Agregar fecha a firma
  - \* Clasroom → Editar el perfil - Todo en mayúsculas y sin acentos
  - la madrugada → El día de entrega siguiente la maestra descargará trabajos y borrar, forceos (links)
  - TRABAJOS NO EN TABLÓN

- \* WhatsApp → Editor perfil
- \* Carpeta de evidencias: (Apuntes) PDF
- \* Todos los trabajos tendrán nomenclatura especial (NOMBRE)

## EXAMEN DIAGNÓSTICO

- ① ¿Qué es un conjunto? Es un grupo o una cierta cantidad de características o elementos.
- ② ¿Cuál es el uso de los conjuntos o utilidades en tu carrera?

Almacena elementos

- ③ ¿Qué es un arreglo y cuál es la función de los arreglos?

Un arreglo es una estructura la cual almacena valores el cual lo hace de una manera más ordenada

- ④ ¿Qué es un compilador?

Es el encargado de pasar todo el código de la maquina a algo entendible por el usuario

- ⑤ ¿Qué es un intérprete?

Es el que ejecuta el código directamente el código fuente de prog. sin haber compilado

- ⑥ ¿Qué es la sintaxis en lenguajes de programación?

Es la forma de escribir el lenguaje para que sea entendible por el computador

- ⑦ ¿Cuál es el objetivo de definir las estructuras a través de su sintaxis?

Para poder organizar y relacionar entre si las unidades del sistema

\* ① Conjunto: En un conjunto no se puede repetir elementos pero en varios conjuntos si se pueden repetir  
 - Intersección, Unión

\* ② Estructura de datos, arrays

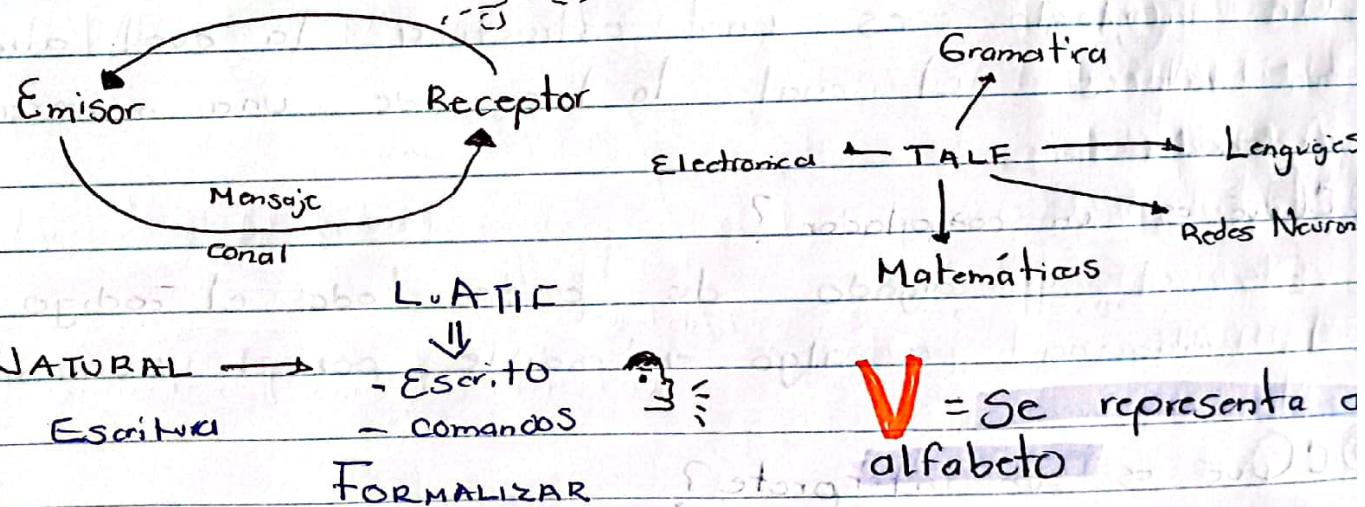
\* ③ CONJUNTOS: Almacena datos

\* ④ COMPILADOR:

⑤ INTERPRETE:

06-09-24

## 1.1 ALFABETO



⑥ Un alfabeto es un conjunto finito y no vacío de elementos llamados símbolos o letras (tiene un fin).

⑦ PALABRA: Una palabra o cadena sobre un alfabeto V (se representara así) es una cadena finita de símbolos del alfabeto cuya longitud indica el número de letras que aparecen en el doble "W".

$|w|$  Longitud de palabra o cadena

Una palabra que no tiene símbolos o letras y cuya longitud es "0" se llama "PALABRA VACIA"

REPRESENTACIÓN

$\lambda$

④ Si  $V$  es un alfabeto, llamaremos " $V^*$ " al conjunto de todas las palabras de longitud  $n$  sobre  $V$  ( $n \in \mathbb{N}$ )

⑤ POTENCIA DE UNA PALABRA: Es una operación que consiste en concatenar la palabra consigo misma " $n$ " veces.

Ejemplo:

$V = \{a, b\}$  → palabras con estas letras

$$w^0 = \lambda \quad w^1 = abaaba$$

$$w^2 = aba \quad w^3 = \underline{aba} \underline{aba} \underline{aba}$$

Recursividad ...

-aba  
-baba  
-ba  
-a  
SP

-abababa

### ⑥ L.3 LENGUAJES FORMALES:

Llamamos lenguaje sobre el alfabeto  $V$  a cualquier subconjunto de  $V^*$ .

Puesto que un lenguaje es tan solo un ~~conjunto~~  $* = \text{estrella Kleene}$  que especial de conjunto. Podemos especificar que un elemento es un lenguaje finito por extensión enumerando puede repetir de los elementos entre llaves. O a mas veces

• Extensión: cuando conozco todas las palabras del lenguaje.

$$L = \{a, aba, baba, b\}$$

Cuando los lenguajes son infinitos los definimos por comprensión y tienen la siguiente estructura:

#### ⑦ COMPRENSIÓN:

$$L = \{w \in V^* \mid w \text{ cumple con la propiedad } P\}$$

El lenguaje es conjunto de cadenas que pertenecen a un alfabeto tal que " $w$ "

$$V = \{0, 1\}$$

$$L = \{w \in \{0, 1\}^* \mid w(\text{ceros}) = w(\text{unos})\}$$

Conjunto de los algoritmos

2

$$\text{Longitud} = |w|$$

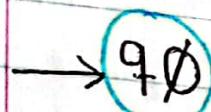
$$E = \text{pertenece}$$

Generen el "lenguaje por comprensión" que acepte donde alfabeto sea  $\{0,1\}$  las palabras con longitud 3

$$V = \{0,1\}$$

$$L = \{wf \mid \{0,1\}^* \mid |w|= \text{sea igual a } 3\}$$

ESTADOS:



INICIAL



ACEPTACION FINAL

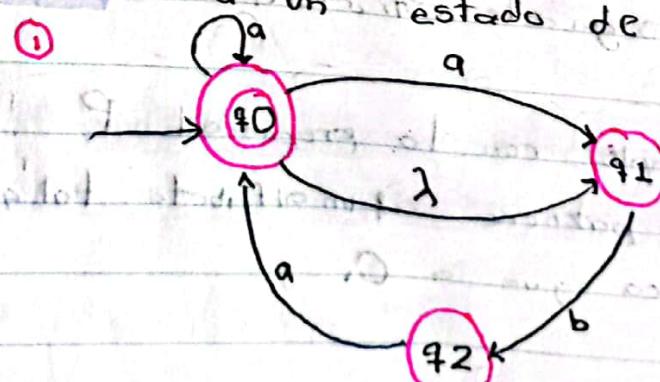
Arcos o flechas (comunicar)

Símbolos (sobre las flechas)

\* EJEMPLO DE DIAGRAMA DE TRANSICIÓN:

Mi lenguaje estaría conformado por todas las palabras que son aceptadas en el diagrama "PALABRAS ACEPTADAS" = VALIDAS. Palabras que se forman del estado

INICIAL a un estado de ACEPTACIÓN



camino 1: aabq

camino 2: aaqaaba

camino 3: ba

$$L = \{ \text{palabras aceptadas} \}$$

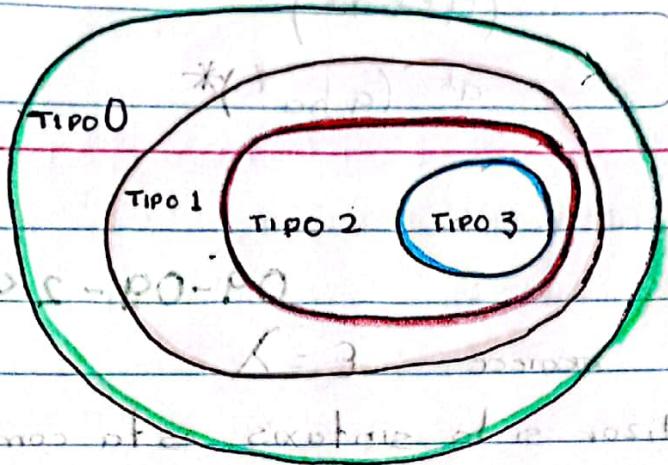


Foman del est. inicial a un est. de aceptación

⑥ Ejemplo de palabras:

$$\begin{array}{l} \textcircled{1} a^* b \\ \textcircled{2} a b^* \end{array}$$

$$\textcircled{1} (ab)^*$$



PS - PO - PA

Tarea: Mapa organizando con todos los conceptos y elenco de esta unidad (Mapa mental) en el cuaderno.

TITULO : Nombre de la Unidad 1

$(a^* b a^*)^*$

① λ

② ab

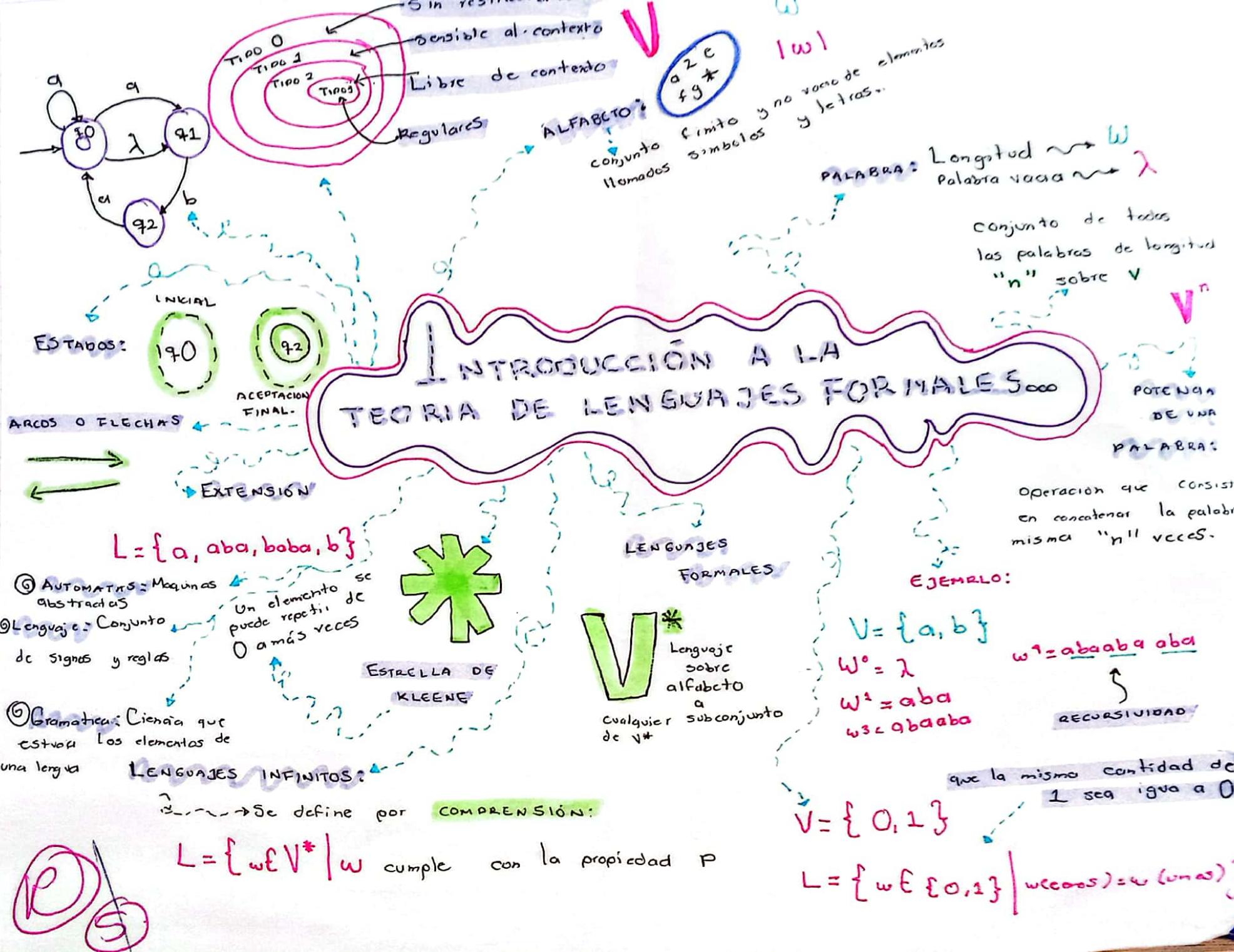
③ No aceptadas

(estantes a = ya que la b es oblig dentro de la secuencia)

④ aa = No tiene la b

y toma la misma cond de aquí

⑤ bb = No existe la g



18-09-24

U

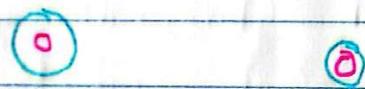
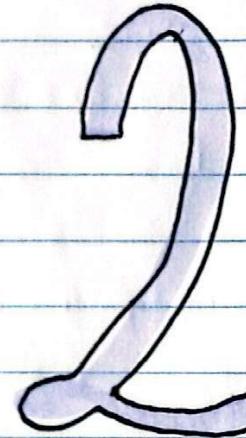
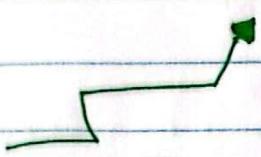
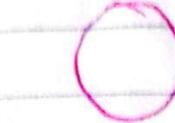
N

I

D

A

D



EXPRESIONES

REGULARES



## CONCEPTOS BÁSICOS SOBRE TRADUCTORES

(10101)

### Lenguajes de primera generación

\* Lenguajes para resolver problemas para que los computadores resolvieran problemas.

para que los ordenadores resolvieran problemas.

Lenguaje binario (se enciende 1 y 0)

en esta etapa era difícil y problemática

aunque después se hizo el uso de código octal o hexadecimal.

### Reemplazado

### Lenguajes de segunda generación

O lenguajes ensambladores

Permite utilizar abreviaturas nómicas como nombres simbólicos

Desventajas:

- Dependencia de la máquina.
- Pocos legibles

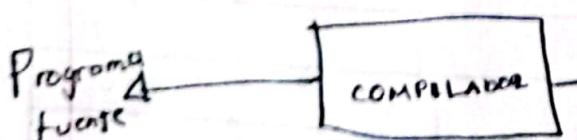
### Tercera generación: ALTO NIVEL (Java, PHP, Python, Javascript, C#)

### Compilador

Lee un programa fuente → lo traduce en un programa durante el proceso de traducción

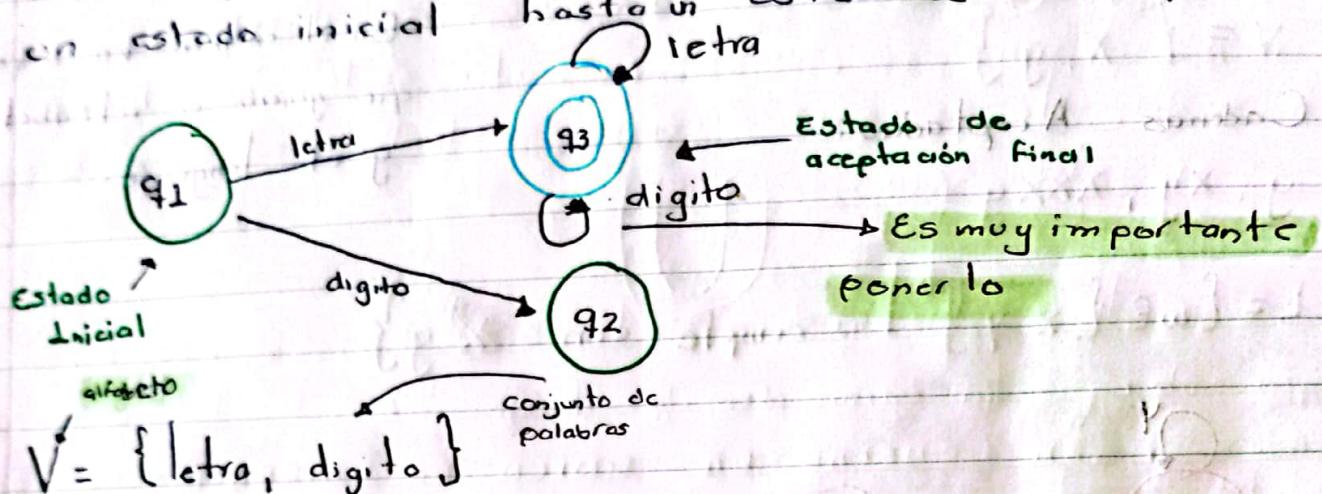
Si presentan los errores en el traducción

programa  
Objetos lógicos



→ Objetos

**CADENA ACEPTADA:** Una cadena aceptada es una cadena de símbolos que se forma al transitar por los flechas del diagrama de transiciones, siempre se formará desde el estado inicial hasta un estado de aceptación.



$$\Sigma = \{\text{letra, digito}\}$$

también  $\Sigma$

$$\text{letra} = \{[a-z]\}$$

$$\text{digito} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L = \{w \in \{\text{letra, digito}\}^* \mid w \text{ cumple con letra (letra* digiton*)}\}$$

→ Lo mas importante para poner

en el examen es lo de color verde

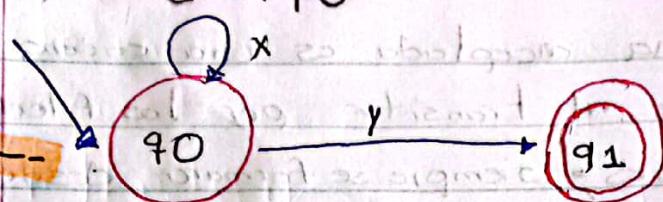
CONCATENACIÓN: "•" (punto)

DEFINICIÓN FORMAL DE:  $\Sigma^*$

**EXPRESIÓN REGULAR:** Una expresión regular (para un alfabeto  $\Sigma$ ) se define formalmente como sigue:

- $\emptyset$  es una expresión regular
- Cada miembro de  $\Sigma$
- Si  $p$  y  $q$  son expresiones regulares, también lo es  $(p \cup q)$
- Si  $p$  y  $q$  son expresiones regulares también  $(p \cdot q)$
- Si  $p$  es una expresión regular también lo es  $p^*$

$$V = \{x, y\}$$



Acepta el lenguaje que consta en cero o más y seguidas por una sola y.

$$V = \{x, y\}$$

Cadenas Aceptadas

$$y, xy, XXXy$$

$$L = \{w \in \{x, y\}^* \mid w \text{ cumple con } x^* y\}$$

En el examen va a venir que generen el alfabeto y el lenguaje.

$$V = \{y, x\}$$

Cadenas Aceptadas

$$x, y x, yy, yx$$

$$V = \{w \in \{x, y\}^* \mid w \text{ cumple con } ly^* x\}$$

Acepta el lenguaje que consiste en cero o más y seguidas por una sola

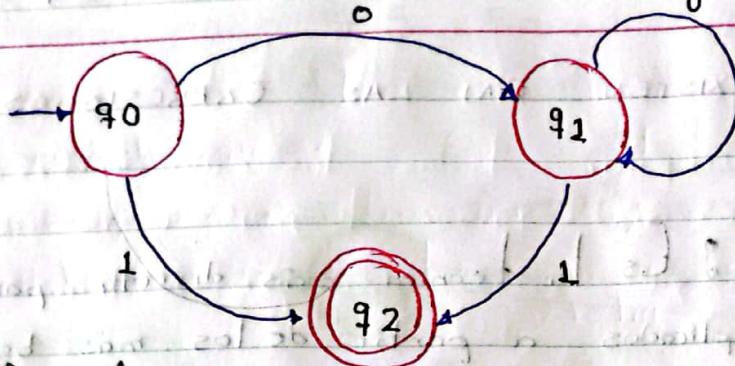
$$\{P, 8, X, J, B, P, E, S, L, O\} = \text{etiquetas}$$

No existen bloques de palabras vacias.

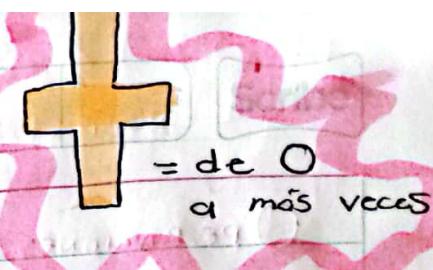
#### L-4 TABLA DE TRANSICIONES:

	X	T	Y	
→ q0	q0		q1	
↓ q1				

p. 100 0\*



$$V = \{1, 0\}$$



Cadenas aceptadas:

1, 001, 01, 001, 0001

$$V = \{w \in \{0, 1\}^* \mid w \text{ cumple con } 0^* 1\}$$

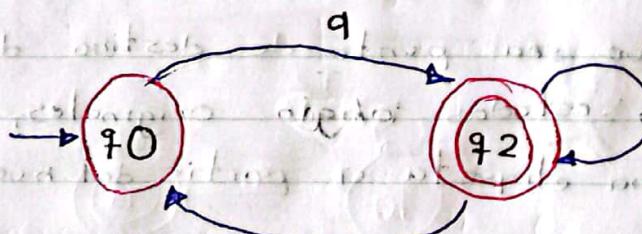
Aquí ya no se pone 1 porque sería más grande el camino y se quita entonces el 1 además adelante sigue el 1

$0^*$   
0  
001  
0001  
00001

0  
1  
001

TABLA DE TRANSICION		
	0	1
q0	q1	q2
q1	q2	q2
q2		

Aquí es cuando sale una flecha para que genere 0 y 1.



a	b
q0	q2
q2	q0

$$V = \{w \in \{a, b\}^* \mid w \text{ cumple con } a^+ (ba^+)^*\}$$

Aquí cumple ya que \* (cero o más veces) puede ir 'a'!

a, aab, aaa, aaaa, aaaaaa

La siguiente cadena puede ser

- ⑥ Es una fórmula que nos permite saber las cadenas aceptadas a través de un alfabeto. Saber las cadenas que comienzan por 'a' y terminan con 'a'. aba, aaa, baaa, abababababa

## OPERACIONES QUE EXISTEN EN LAS EXPRESIONES REGULARES

**UNION:** Es la técnica más directa para combinar lenguajes más complicados a partir de los más básicos utilizando la operación de unión de la teoría de conjuntos y se representa por: **U**

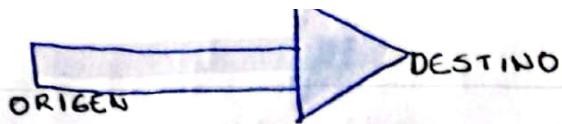
Para saber si la unión de dos lenguajes

en general la UNION DE DIAGRAMAS DE TRANSICIÓN:

- 1: Se dibuja un nuevo estado inicial.
- 2: Se declara a este nuevo estado inicial como estado final si y solo si uno de los estados iniciales anteriores era de aceptación.
- 3: Para cada estado que sea punto de destino de un arco de algunos de los estados iniciales originales, dibuja una flecha con la misma etiqueta a partir del nuevo estado inicial.
- 4: Elimina la característica de inicio de los estados iniciales originales.

### PUNTUALIZACIONES:

- Solo pueden tener un estado inicial en los diagramas regulares.
- Puede haber más de un estado final.
- Al estado final también se le llama estado de aceptación.
- Una flecha también se conoce como arco.
- Una flecha puede ser origen o destino.



ST Q LT

23 sept. 24

Scribe

El ultimo paso #5 Renombra los estados

Paso 1

Paso 3

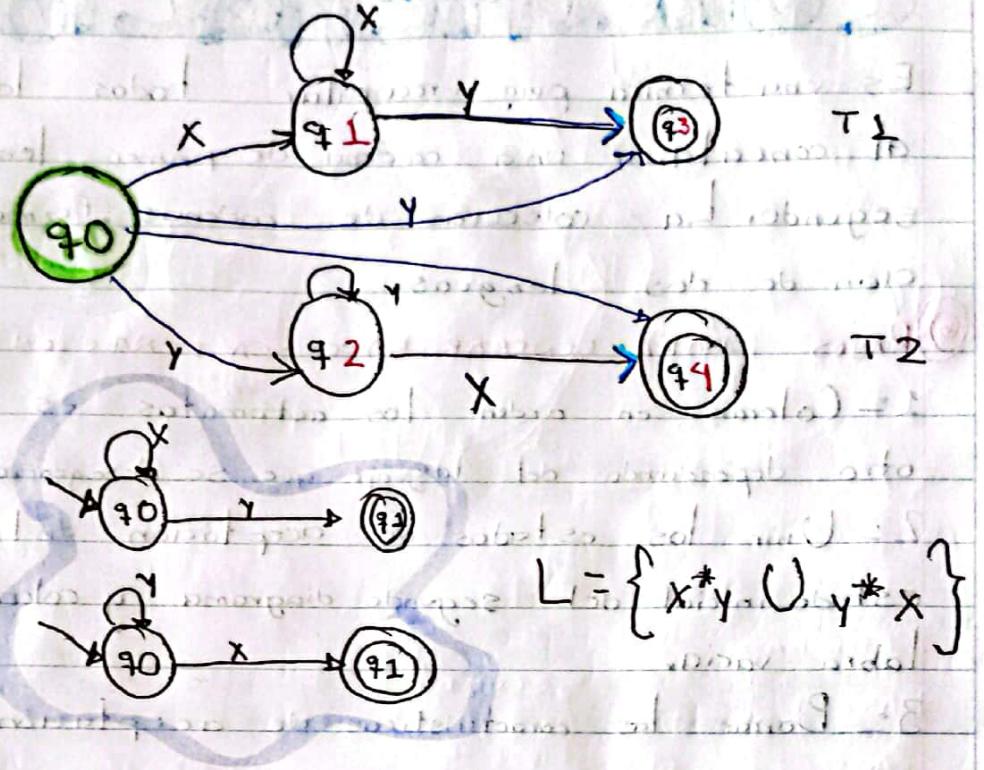
Paso 5 renombra los estados

Nota: El paso número

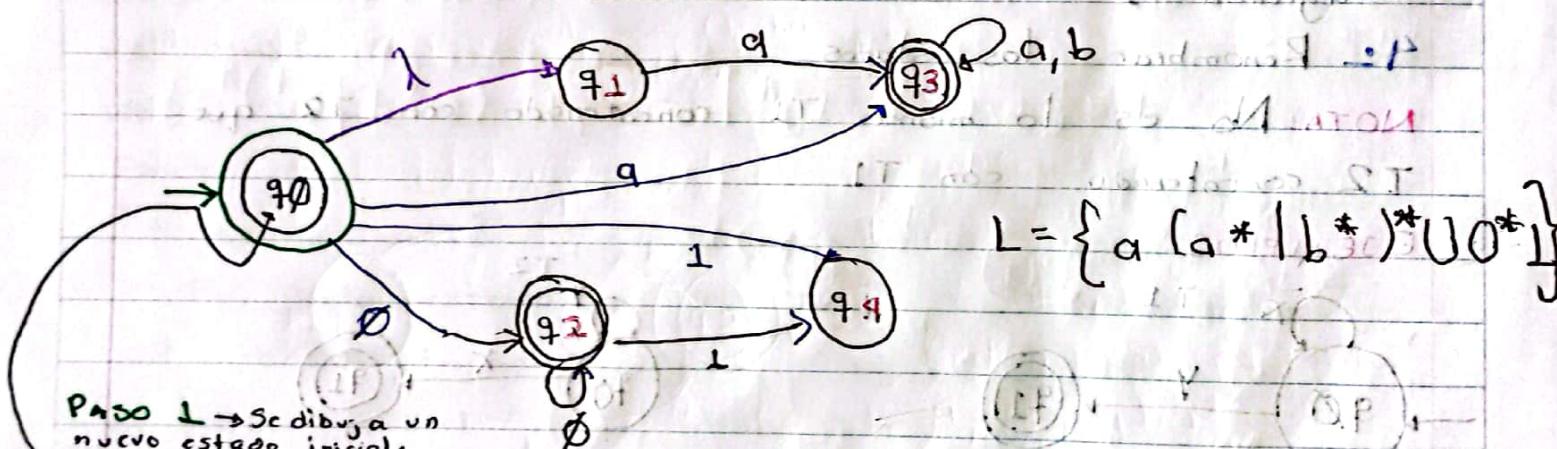
2 y 4 no aplican en este ejemplo

Se utilizo estos

2 diagramas originales



$$L = \{x^*y \cup y^*x\}$$



Paso 1: Se dibuja un nuevo estado inicial.

Paso 2: Se dellara estado

inicial a q0 solo si adelante tiene uno igual

Paso 3: dibujar las flechas en los estados

iniciales  $\rightarrow$

Paso 4:

Paso 5: Renombra estados

$$\{(x+y) \circ (y^k)\} \text{ no es igual a } \{x^k, y^k\} \text{ para } k \geq 1$$

Porque

Escaneado con CamScanner

T1

T2

concatenación

OUT 100

25 09 24

# Concatenación:

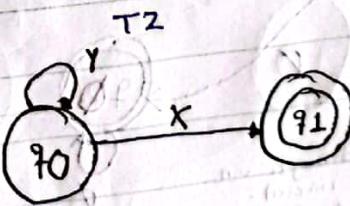
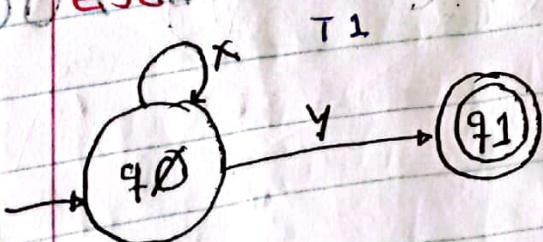
Es una técnica para recopilar todas las cadenas formadas al concatenar una cadena de primer lenguaje y una cadena segundo. La colección de cadenas formadas se llama concatenación de dos lenguajes.

## 6 PASOS PARA GENERAR LA CONCATENACIÓN:

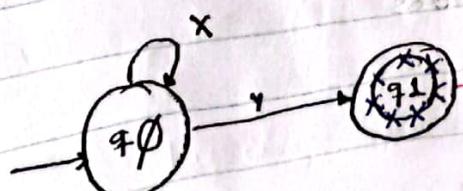
- 1 - Colocar en orden los automatas es decir uno enfrente otro dependiendo del lenguaje que se genera
- 2 - Unir los estados de aceptación del primer diagrama con el estado inicial del segundo diagrama y colocar la etiqueta de labra vacía.
- 3 - Borrar las características de aceptación del automata del diagrama 1
- 4 - Renombrar los estados

**NOTA:** No es lo mismo T1 concatenado con T2 que T2 concatenado con T1

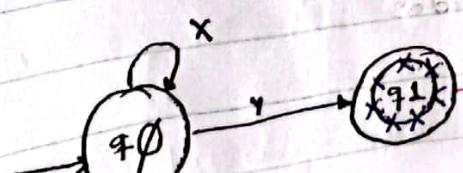
## EJEMPLO:



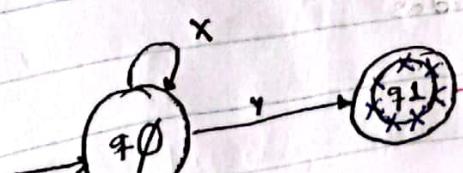
Paso 1



Paso 2



Paso 3

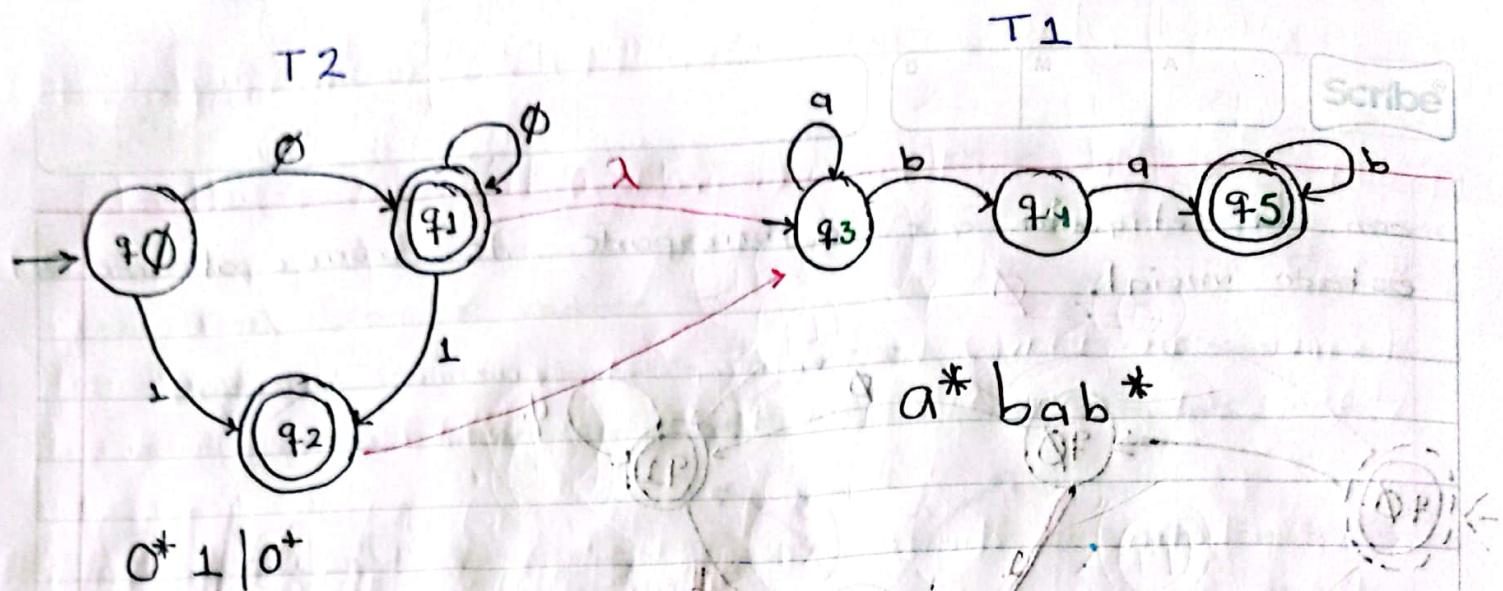


Paso 4

$$L = \{w \in \{x, y\}^* \mid w \text{ cumple con } (x^*y)^* \circ (y^*x)^*\}$$

T1 concatenado con T2  
Para poder concatenar el orden de T2 a T1 se cambia el orden de T2

T1 concatenado con T2  
Para poder concatenar el orden de T2 a T1 se cambia el orden de T2



Paso 1

$$L = \{ w \in \{a, b, 0, 1\}^* \mid w \text{ cumple con } (0^* 1 | 0^*) \circ (a^* b a b^*) \}$$

Paso 2

Paso 3

Paso 4

### 3. Estrella de Kleene:

Infiere de los anteriores en que amplia un solo lenguaje en lugar de combinar 2, esto se logra formando todas las concatenaciones de 0 (a) más cadenas de lenguajes que se ampliarán.

PASOS:

PRIMERA ETAPA: 1-Dibuja un nuevo estado inicial.

2- Por cada estado que sea punto de destino de una flecha del estado inicial original, dibujamos una flecha con la misma etiqueta desde el nuevo estado inicial.

3- Designar a este nuevo estado como de aceptación.

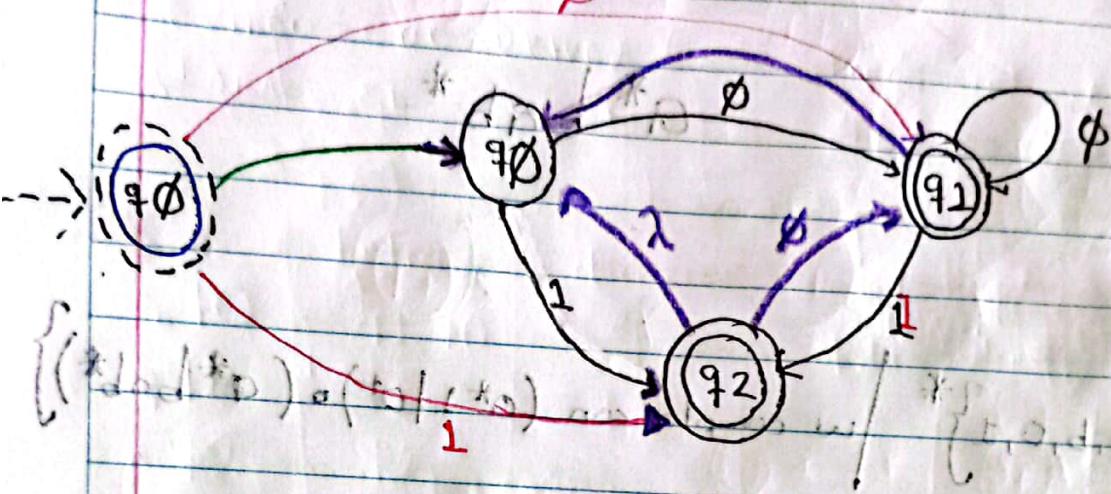
4- Verificar que no exista más de un estado inicial en el diagrama final, en caso de existir, concatenar el nuevo estado inicial con los estados concatenados iniciales y colocar la etiqueta que le corresponde.

SEGUNDA ETAPA:

5- Añade una flecha de cada estado de aceptación a cada estado que es el destino del estado inicial.

6- Cada uno de estos nuevos arcos se rotula

con la etiqueta que le corresponde de acuerdo al arcón inicial.



PRIMERA:

PASO 1

SEGUNDA

PASO 5

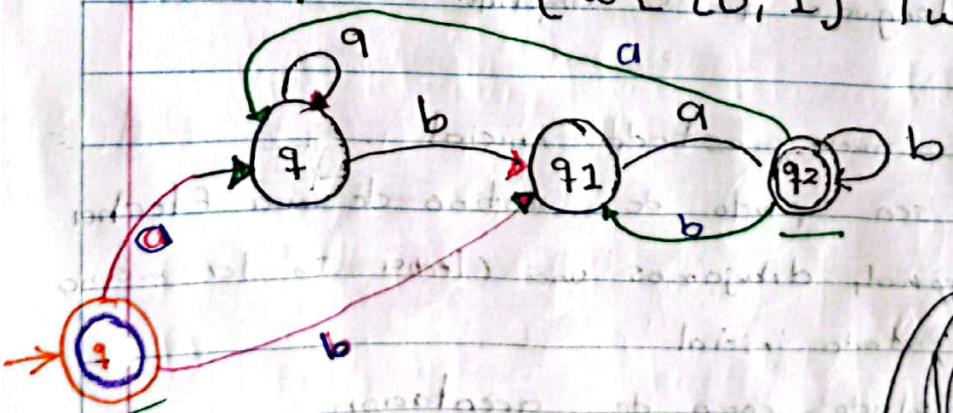
PASO 2

PASO 6

PASO 3

PASO 4

$$L = \{ w \in \{0, 1\}^* \mid w \text{ simple } (0^* 1 | 0^+)^* \}$$



PRIMERA:

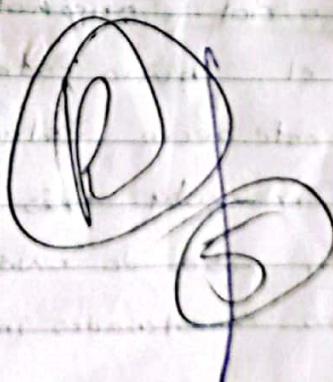
PASO 5

PASO 1

PASO 6

PASO 2

PASO 3



## 2.3 APLICACIONES REALES



a) COMPILADORES: Ya que un compilador debe ser capaz de reconocer cual son los cadenas de simbolos del programa fuente por ejemplo los nombres de las variables, las constantes numericas y las expresiones regulares, palabras reservadas

b) EN INTELIGENCIA ARTIFICIAL: Las máquinas de estado finito tienen un gran futuro en la inteligencia artificial, su principal objetivo es la creación de una agente inteligente que sea capaz de actuar e interpretar instrucciones de manera muy parecida a como lo hacemos los humanos. Para la creación de este agente es necesario contar con un total reconocimiento de la gramática y los lenguajes formales, para ello se debe contar con una máquina que sea capaz de aceptar simbolos y reconocer las cadenas que se están usando.

c) EN ELECTRÓNICA: A través de las máquinas de estado finito como aceptadores de simbolos se aplica en la electrónica para la creación de circuitos, ya que tienen que reconocer que una cadena de simbolos es idéntica a otra, y así dejar pasar los datos, por ejemplo en semáforos circuitos integrados, apagadores automáticos de luz, sistemas automáticos de riego, entre otros.

### ⑥ Concatenación

INDICES: Alfabeto, lenguaje por comprensión, diagrama de transiciones  
tabla de trasmisiones

## Manual 2 de Octubre (miércoles).

Duración máxima 5 minutos

EQUIPO: (video de DRIVE). (se entrega el próximo miércoles)  
(PARA EL VIERNES 04 DE OCTUBRE)

T1

Letra	dígito	pesos	2
→ 90	91, 90	93	91
91	91	92	91
92	92	93	93
93	93	93	93

T2

Létra-Mays	Létra-Minus
→ 90	91
91	92
92	92
93	93

T3

Signo	dígito	(aritmético)	operador	igual	2
→ 90	91				
91	92				
92	92				
93					
94					
				H1	90

Scrible

T4					
digito	punto	operador	igual	=	
→ q0	q1				
q1	q1	q0	q2	q3	
q2					
↓ q3					q0

\* = estrella de klein

Operaciones :

UFunión

◦ = concatenado

T1 ∪ T2

T2 ∪ T4

T3 ∪ T4

T1 ◦ T2

T3\*

T2 ◦ T1

T1\*

CONTENIDO :

desarrollo

⑥ Manual: (Alfabeto, elementos del alfabeto, conjuntos, expresión regular, lenguaje por compresión, → diagrama y tabla de transiciones, operación y luego ↑)

⑦ Vídeo:

① Presentación (tema, objetivo)

② Generación del alfabeto y conjuntos

③ Diagrama de transiciones de manera individual (T1 y T2) expresión regular individualmente

④ Pasos para hacer la operación que nos toca e ir dibujando lo resultante

⑤ Lenguaje generado

⑥ Tabla de transiciones resultante de la operación

⑦ Audio (que nos veamos nosotros)

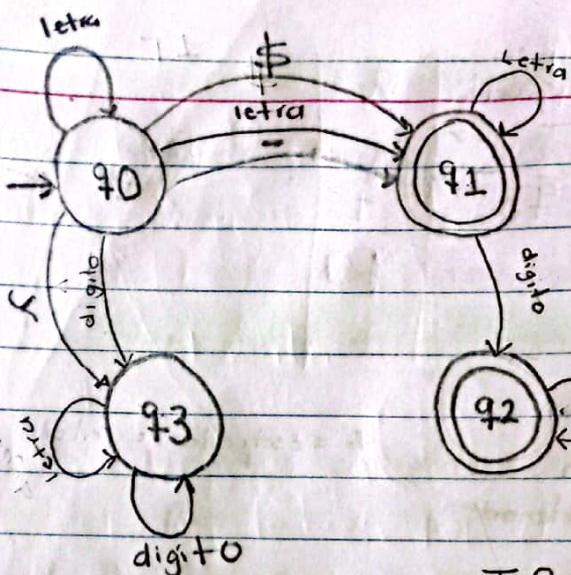
⑧ Duración no más de 10 minutos.

- = guion bajo

T 1

$$V = \{ \text{letra, dígito, } \$, -, ^\wedge \}$$

$$\text{letra} = \{ [a-z] \} \quad \text{dígito} = \{ [0-9] \}$$



Cadenas aceptadas:

$\$, \text{letra}, \text{letra letra letra}$

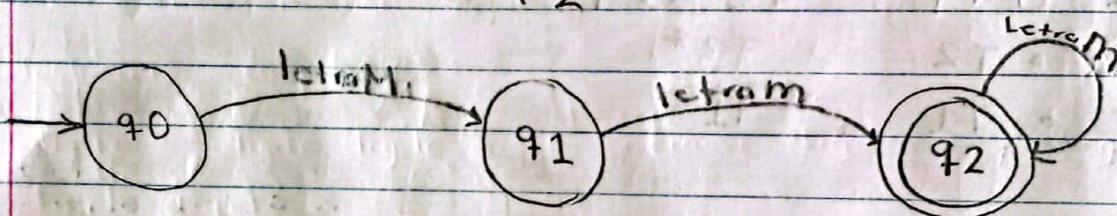
Scribe

$$L = \{ w \in \{ \text{letra, dígito, } \$, -, ^\wedge \}^* \}$$

$$L = \{ w \in \{ \text{letra}^*, \$ - (\text{letra}^* | \text{dígito}^* | \wedge^*) \}^* \mid \text{completo} \}$$

$$( \text{letra}^* ) \vdash ( \text{letra}^* | \text{dígito}^* | \wedge^* ) \vdash \text{letra}^*$$

T 2



$$V = \{ \text{letraM}, \text{letraM} \}$$

complemento de T1:  $\text{letraM} = \{ [a-z] \}$ , igual

Cadena aceptada:

M mm m M M m m m M m m m m m m

⑥ Lenguaje por comprensión:  $(\text{letraM}, \text{letraM})^*$

$$L = \{ w \in \{ \text{letraM}, \text{letraM} \}^* \mid w \text{ es completo} \}$$

letraM letraM letraM

M m m

M m m m

M m

m m m

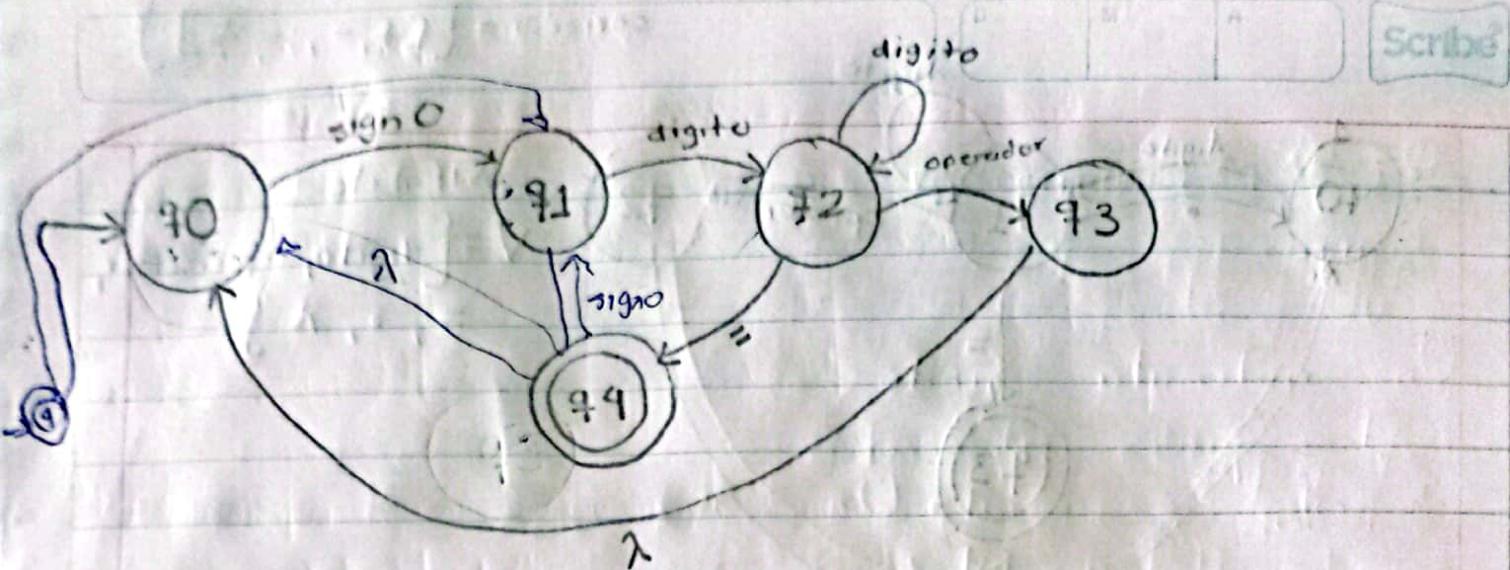
obras enemigo

continua al 20

continua con el 20

continua con el 20

T3 | P-02] dígito = { [0-9] }    aritmético = { [+, -, ×, ÷] }



$$V = \{ \text{signo, dígito, operador aritmético, } \lambda \}$$

cadenas aceptadas: signo dígito, signo dígito operador dígito, signo dígito operador signo dígito, signo dígito operador signo dígito dígito

Lenguaje por comprensión:

$$L = \{ w \in \{ \text{signo, dígito, operador aritmético, } \lambda \}^* \mid w \text{ comple}$$

$$( \text{signo} ) \text{ dígito}^+ \text{ operador dígito}^+ = \}$$

↑  
Al menos necesito  
un dígito ya sea  
normal o bucle

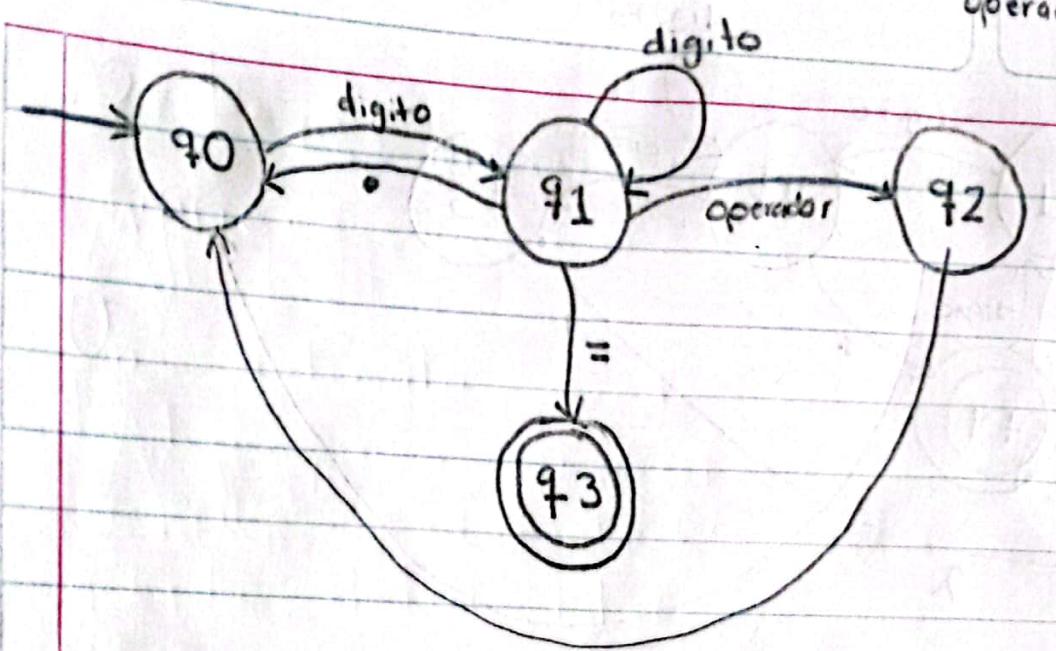
$$\triangleright \text{signo dígito}^+ = , \text{ signo dígito operador} =$$

$$+/- 33 + 4 =$$

$$+/- 1 + =$$

digito = { [0-9] }

operador = { [ +, -, \*, / ] }



Alfabeto :  
 $V = \{ \text{digito}, \text{operador} \}$   
 $=, +, -, *, / \}$

Cadenas aceptadas :

digito = , digito digito digito operador digito = , digito + digito  
digito digito =

Expresión :

digito

digito<sup>+</sup> punto digito<sup>\*</sup> operador digito<sup>+</sup> punto digito<sup>\*</sup>

$$3 \cdot 3 + 3 \cdot 3 =$$