

LENGUAJES

Y AUTÓMATAS

BCD 2-3-5

MTI. Yadira Esther Jiménez Pérez

• COMPETENCIA:

Define, diseña y programa las fases del analizador léxico y sintático de un traductor o compilador para preambulos de la construcción de un compilador.

UNIDAD 1

INTRODUCCIÓN A LA TEORÍA DE LENGUAJES FORMALES

1. INTRODUCCIÓN A LA TEORÍA DE LENGUAJES FORMALES

1.1. Alfabeto

1.2. Cadenas

1.3. Lenguajes: tipos y herramientas

1.4. Estructuras de un traductor

1.5. Fases de un compilador

2. EXPRESIONES REGULARES (ER)

2.1. Definición formal de una expresión regular

2.2. Diseño de una expresión regular

2.3. Aplicaciones en problemas reales

3. AUTÓMATAS FINITOS (AF)

3.1. Conceptos: Definición y clasificación de un autómata finito

3.2. Conversión de un autómata finito no determinístico (AFN) a un autómata finito determinístico (AFD)

3.3. Representación de una expresión regular usando autómatas finitos no determinísticos

3.4. Minimización de estados de un autómata finito

3.5. Aplicaciones

4. ANÁLISIS LÉXICO

4.1. Funciones del analizador léxico

4.2. Componentes léxicos, patrones y léxemas

4.3. Creación de la tabla de tokens

4.4. Errores léxicos

4.5. Generadores de analizadores léxicos

4.6. Aplicaciones

5. ANÁLISIS SINTÁCTICO

- 5.1. Definición de clasificación de gramáticas.
- 5.2. Gramáticas libres de contexto (GLC)
- 5.3. Algoritmos de derivación.
- 5.4. Forma normal de Chomsky
- 5.5. Diagramas de sintaxis
- 5.6. Eliminación de la ambigüedad.
- 5.7. Tipos de analizadores sintácticos
- 5.8. Manejo de errores
- 5.9. Generadores de analizadores sintácticos.

6. MAQUINAS 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 por la máquina de Turing.

BIBLIOGRAFIA:

- Aho, Alfred (2007). Compiladores, Principios, técnicas y herramientas. México: Pearson Educación.
- Hopcroft, John (2002). Introducción a la teoría de autómatas, lenguajes y computación. Madrid: Addison-Wesley.
- Kelley D (1995). Teoría de Autómatas y Lenguajes Formales. Madrid: Prentice Hall.
- Martín, J (2004). Lenguajes Formales y Teoría de la Computación. México McGraw Hill / Interamericana de México.

INTERNET

- Teoría de Automatas y Lenguajes Formales (TALF)
- Teoría de la computación
- Autómatas
- Compiladores

EVALUACIÓN

UNIDAD 1

- Examen Diagnóstico 10 %
- Actividades clase / Tareas 20 %
- Exposición de fases del compilador 30 %
- Examen 40 %

UNIDAD 2 y 3

- Actividades de clase / Tareas 30 %
- Manual de Prácticas de Expresiones regulares 30 %
- Examen 40 %

UNIDAD 4

- Manual del analizador lógico 30%
- Exposición del analizador lógico ... 20%
- Proyecto final 50 %.

UNIDAD 5

- Manual del analizador sintáctico 30%
- Exposición 30 %
- Proyecto final ... 40%.

UNIDAD 6

- Automatización de la configuración de la cinta de una máquina de Turing 30%
- Exposición de ejercicios de la máquina de Turing 30%
- Examín 40%.

F. EXAMENES

14-18 Octubre. → 16 o 17 Octubre.
 → 20 o 21 Noviembre.
 → 15 o 16 Enero.

Recuperación 22 de Enero

Reglas de clase

- Respeto
- Jueves → Tolerancia 10 minutos.
- Entradas classroom → Fechas.
- Nomenclatura al archivo.
- Whatsapp → Inb grupo → liga classroom.
- Whatsapp / Classroom → Nombre y Apellido. → Todo en mayúsculas y sin espacios

EXAMEN DIAGNÓSTICO

D M A
3501. 04 Septiembre 2024

Scrib

1. ¿Qué es un conjunto?

- Es una agrupación de objetos ó datos e incluso de elementos que puedan tener características en común.

2. ¿Qué operaciones de conjuntos recordás?

- Unión (U), Intersección (I) y Diferencia.

3. ¿Qué es un arreglo?

- Es una estructura de datos en la cual podemos almacenar datos y asignarles un índice para después poder hacer consultas de esta manera.

4. ¿Cuál es la función de un arreglo?

Almacenar datos para posteriormente realizar consultas, hacer modificaciones e incluso operaciones.

5. ¿Cómo o de qué está conformada una cadena?

Por un conjunto de caracteres. Los cuales pueden ser letras, números, espacios o símbolos.

6. ¿A qué se refiere la sintaxis de un lenguaje?

Son un conjunto de reglas para escribir estructuras y generar expresiones con sentido, es decir el como se van a escribir las instrucciones para generar acciones, estas reglas son propias para cada lenguaje.

7. ¿Cuál es el objetivo de conocer la sintaxis de ciertas estructuras en un lenguaje de programación?

El saber como funcionan y entender su comportamiento, para una buena escritura de código para que este sea comprendible para otros programadores y para analizar que estructura se va a ocupar dependiendo del problema que se presente.

8. ¿Qué es un compilador?

Es un programa que traduce el lenguaje de programación al lenguaje máquina para que se ejecuten las acciones planeadas.

9. ¿Qué es un intérprete?

Es un programa que analiza el código linea por linea, interpreta su significado y ejecuta las acciones que lo corresponden.

10. ¿Qué es un traductor?

Es un programa que nos permite pasar de un lenguaje de programación a otro.

CONJUNTO → Tienen un nombre
Características en común.

Ejemplo: Un arreglo
Bases de datos

OPERACIONES → Unión
Intersección
Diferencia
Complemento

BASE DE DATOS → Operaciones
Join

→ Tarea: Buscar.

Relación de la base de datos con conjuntos. (Diagrama de Venn)

ARRREGLO → Almacenar datos del mismo tipo y manipularlos con el mismo nombre.
como está CONFORMADA UNA CADENA → conjunto de caracteres

SINTAXIS → Estructura, orden.

COMPILADOR → Traduce código de alto nivel a un ensamblador o un lenguaje máquina.

INTERPRETE → Traduce / ejecuta línea por línea

DIFERENCIAS INTERPRETE y COMPILADOR: Ambas son traductoras, pero el compilador lo hace en una sola toma y genera un archivo, y el interprete lo hace línea por línea
traduce / ejecuta.

Un compilador traduce todo el código al principio y todo en bloques
de un tiempo igual. Una compilación del código se hace en bloques
y cada uno es más eficiente que el otro.

Un intérprete traduce línea por línea el código a medida que se lee.

LENGUAJES Y AUTÓMATAS

D M A
05 Septiembre 2024

Scribem

Leng. Alto Nivel

Traducción \Rightarrow COMPILADOR

FASES DE ANÁLISIS:

1 Léxico

↓

Diccionario de todos los elementos que pueda reconocer el lenguaje

2 Sintáctico

↓

Estructura

- corporal
- señas.
- oral
- escrito.

LENGUAJE NATURAL

- Más formal.
- Es más estructurado.

LENGUAJE MÁRTIFICIAL

• CONCEPTOS BÁSICOS

ALFABETO: Es un conjunto finito y no vacío de elementos. llamados símbolos o letras. $V(V)$

CADENA: Una palabra o cadena sobre un alfabeto (V) es una cadena finita de símbolos del alfabeto. La longitud de una cadena (w) la denotaremos como $|w|$ que indica el número de letras o símbolos que aparecen en w .

N.T: Cuando la longitud de una palabra es 0, no tiene símbolos, la conocemos como palabra vacía λ .

Longitud \emptyset = Palabra vacía

Si V es un alfabeto: llamaremos V^n al conjunto de todas las palabras de longitud n sobre V y llamaremos V^0 al conjunto cuyo único elemento es la palabra vacía.

$$V^0 = \{\lambda\}$$

El conjunto de todas las cadenas de cualquier longitud sobre V tiene que estar siempre representado para indicar el lenguaje que se formará con él mismo; de esta manera llamamos potencia de una palabra a la operación que consiste en concatenar la palabra n veces consigo misma y se representa de la siguiente manera

$$w \in V^*$$

* Estrella de clin

Se puede repetir de 0 a N veces esa elemento.

→ EJEMPLO

$$V = \{a, b\}$$

$W^0 = \{\lambda\} \rightarrow$ Palabra vacía

$$W^1 = bba$$

$$W^2 = bbabbba$$

$$W^3 = bbabbba$$

Con el alfabeto se crean palabras

\hookrightarrow Pertenecen

\mapsto tal que.

El número de veces en el que se van a concatenar las palabras
↓ las mismas.

PALABRAS

a

ab

aab

bbb

abab

b

aba

Cadena → conjunto de símbolos y letras conformado
a partir de un alfabeto. (elementos)

LENGUAJES FORMALES

Llamamos lenguajes sobre el alfabeto V a aquellos subconjunto de V^* , ya que un lenguaje es tan solo una clase especial de conjuntos y podemos especificarlo de dos maneras:

① Por Extensión: Se define todos y cada uno de los elementos, enumerando los elementos entre llaves, cuando los lenguajes son finitos.

LENGUAJE: Conjunto de cadenas basadas en un alfabeto que permiten comunicar algo.

Por extensión \Rightarrow Finitos. (Se conoce exactamente el número de palabras que se puedan formar).

Ej. las palabras racionales

$$L = \{for, if, else\}$$

② Por Comprendión: Se emplea una premisa o una propiedad que se debe cumplir para que estas palabras formen parte de un lenguaje y se define de la siguiente forma:

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

$$V = \{0, 1\} \quad L = \{ w \in V^* \mid \#0(w) = \#1(w) \}$$

$$w_0 = w_1$$

Defina por comprensión el lenguaje formado por aquellas cadenas que tienen la misma cantidad de 0 y 1.

• Lenguaje es igual a = quiero obtener cadenas que pertenezcan al alfabeto 0 y 1, donde 0 y 1 pueden aparecer n veces, tal que la cadena de ceros es igual a las cadenas de unos

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

Lenguaje que me permita apartir del alfabeto $V = \{0,1\}$ crear palabras de longitud 3.

$$L = \{ w \in \{0,1\}^* \mid |w| \leq 3 \}$$

$$V = \{a,b\}$$

a^* → Estrella da clin → Indica los elementos que se puedan repetir, desde 0 hasta más veces.

—, a, aa, aaa, aaaa...

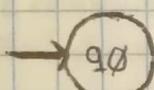
$$ab^*$$

—, ab, abb, abbb, abbbb.

$$(ab)^*$$

—, ab, abab, ababab, ababab.

DIAGRAMAS DE TRANSICIÓN



Inicial



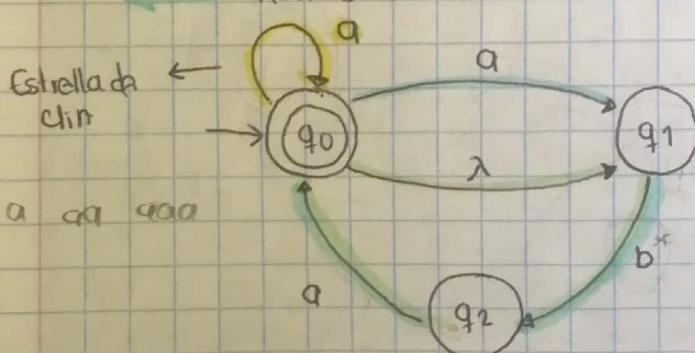
Final
Aceptación

NOTA:

El estado inicial también puede ser un estado de aceptación (final).

ESTADOS

Arcos } Marcan el sentido de los estados/transiciones.
Flechas }



$$V = \{a,b\}$$

La palabra vacía no se coloca en el alfabeto

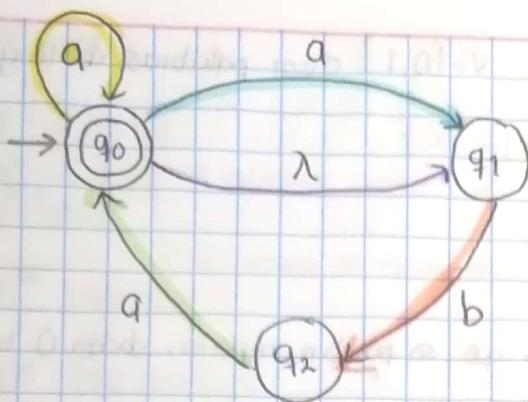
ba
aba }
Palabras
aceptadas

Mientras se sigan las flechas todas las palabras son válidas.

El lenguaje debe estar conformado por todos los palabras aceptadas.

En un lenguaje de programación nos ayuda a saber qué identificadores son válidos para una variable, función, método, clase, constante.

a, a*, a*ba*, a*ba, aba*, ba*, ba*



$L = \{aba^*, a^*ba, a(ba)^*\}$
 $(a^*ba)^*$

a^*
 a^*aba^*
 a^*baa^*
 a^*aba
 a^*ba

λ
 a
 $aaba$
 $abaa$
 $abaa$
 aba

aba
 ba
 $abaa^*$
 baa^*

$abaa$
 ba
 $abaa$
 baa

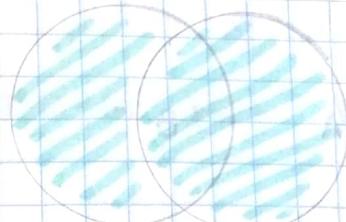
TEORIA DE CONJUNTOS

APLICADA A LAS BASES DE DATOS

UNIÓN

Es una operación en la que se combinan dos o más tablas o conjuntos de datos en una sola tabla o conjunto de datos. La unión se realiza generalmente en función de una o más columnas comunes entre las tablas que se están uniendo.

En términos más simples, la unión de bases de datos se utiliza para combinar información de diferentes tablas en una sola tabla para obtener una visión más completa de los datos.



$A \cup B$

Comprendiendo:

$$A = \{ \text{idusuario, nombre, apellido, cedula, idcargo, idrol} \}$$

$$B = \{ \text{idcargo, descripción} \}$$

$$A \cup B = \{ \text{idusuario, nombre, apellido, cedula, idcargo, idrol, idcargo, descripción} \}$$

Extención:

$$A = \{ x | x = (\text{idusuario, nombre, apellido, cedula, idcargo, idrol}) \}$$

$$B = \{ x | x = (\text{idcargo, descripción}) \}$$

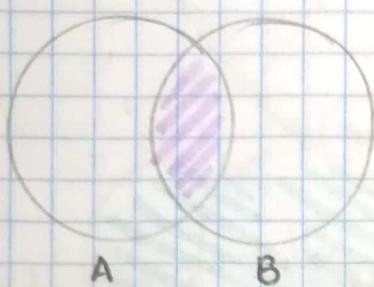
$$A \cup B = \{ x | x \text{ pertenece a } A \vee x \text{ pertenece a } B \}$$

MariaDB [factcom]> select U. *, C. * from ususario U, cargo C;						
idUsuario	nombre	apellido	cedula	idcargo	idrol	idcargo
2	andres	salazar	1234567890	1	2	1
2	andres	salazar	1234567890	1	2	2
2	andres	salazar	1234567890	1	2	3

3 rows in set (0.001 sec)

~~P
D~~

INTERSECCIÓN



La intersección en una base de datos es una operación de conjunto que devuelve únicamente los registros que aparecen en dos o más conjuntos de datos. Se basa en la comparación de valores en una o más columnas comunes entre los conjuntos de datos.

La intersección se puede realizar utilizando el operador SQL "INTERSECT" o utilizando una combinación de cláusulas "JOIN" y "WHERE" en una sola consulta SQL.

COMPRENSIÓN

$A = \{\text{Idusuario, nombre, apellido, cedula, idcargo, idrol}\}$

$B = \{\text{id cargo, descripción}\}$

$A \cap B = \{\text{id cargo}\}$

EXTENSIÓN

$A = \{x | x = (\text{Idusuario, nombre, apellido, cedula, idcargo, idrol})\}$

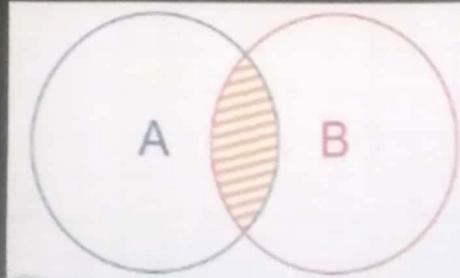
$B = \{x | x = (\text{id cargo, descripción})\}$

$A \cap B = \{x | x \text{ pertenece a } A \text{ y } x \text{ pertenece a } B\}$

```
MariaDB [factcom]> SELECT ususario.idcargo
-> FROM ususario
-> INNER JOIN cargo
-> ON ususario.idcargo = cargo.idcargo;
```

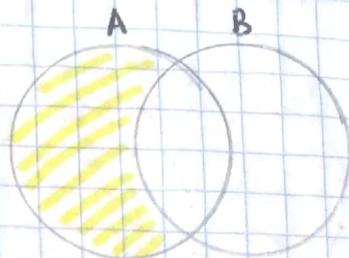
```
+-----+
| idcargo |
+-----+
|      1   |
+-----+
1 row in set (0.104 sec)
```

```
MariaDB [factcom]>
```



DIFERENCIA

La diferencia de conjuntos se refiere a una operación que se utiliza para obtener los elementos que se encuentran en un conjunto de datos y no en otro. En otras palabras, la diferencia de conjuntos es una operación que devuelve los elementos que pertenecen a un conjunto de datos y no están presentes en otro conjunto de datos.



Sintaxis en MySQL: `SELECT * FROM tabla1 WHERE id NOT IN (SELECT id FROM tabla2);`

→ Comprendimiento

$$\begin{aligned} A &= \{ \text{idusuario, nombre, apellido, cedula, idcargo, idrol} \} \\ B &= \{ \text{id cargo, descripción} \} \\ A \setminus B \cup B - A &= \{ \text{idusuario, nombre, apellido, cedula, idrol} \} \end{aligned}$$

→ Extensión

$$\begin{aligned} A &= \{ x | x = (\text{idusuario, nombre, apellido, cedula, idrol}) \} \\ B &= \{ x | x = \text{id cargo, descripción} \} \\ A \setminus B \cup B - A &= \{ x | x \in A \text{ y } x \notin B \} \end{aligned}$$

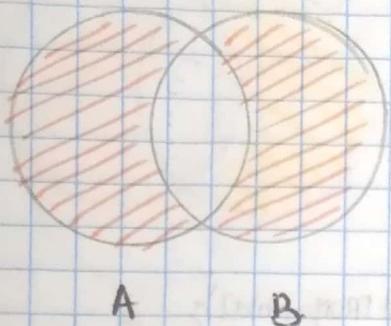
```
MariaDB [factcom]> select * from ususario
```

```
-> where exists (select 1 from cargo
-> where cargo.idcargo = ususario.idcargo);
```

idUsuario	nombre	apellido	cedula	idcargo	idrol
2	andres	salazar	1234567890	1	2

1 row in set (0.001 sec)

DIFERENCIA SIMETRICA



Es una operación que devuelva los elementos que están en uno de los dos conjuntos de datos, pero no en ambos conjuntos de datos.

En otras palabras, es la unión de los elementos que son únicos en cada uno de los dos conjuntos de datos. Matemáticamente la diferencia simétrica entre $A \text{ y } B$ se denota como

$$A \Delta B \quad A \Delta B = (A - B) \cup (B - A)$$

Donde " $-$ " denota la diferencia entre conjuntos. En otras palabras, la diferencia simétrica entre dos conjuntos $A \text{ y } B$, pero no en ambos.

✓ Comprendimiento

$A = \{\text{idusuario, nombre, apellido, cedula, idcargo, idrol}\}$

$B = \{\text{idcargo, descripción}\}$

$(A \cup B) - (A \cap B) = \{\text{idusuario, nombre, apellido, cedula, idcargo, idrol}\} \cup \{\text{idcargo, descripción}\}$

✓ Extensión

$A = \{x | x = (\text{idusuario, nombre, apellido, cedula, idcargo, idrol})\}$

$B = \{x | x = (\text{idcargo, descripción})\}$

$A * B = \{x | x \in A \text{ y no pertenece a } B \text{ o } x \in B \text{ y no pertenece a } A\}$

```
MariaDB [factcom]> SELECT idusuario, nombre, apellido, cedula, idcargo, idrol, NULL AS descripción
```

```
-> FROM usuario
-> WHERE idcargo NOT IN (SELECT idcargo FROM cargo)
-> UNION
-> SELECT NULL AS idusuario, NULL AS nombre, NULL AS apellido, NULL AS cedula, idcargo
, NULL AS idrol, descripción
-> FROM cargo
-> WHERE idcargo NOT IN (SELECT idcargo FROM usuario);
```

idusuario	nombre	apellido	cedula	idcargo	idrol	descripción
NULL	NULL	NULL	NULL	2	NULL	operativo
NULL	NULL	NULL	NULL	3	NULL	auxiliar

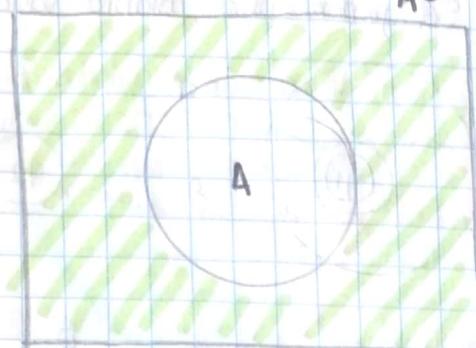
emaze

2 rows in set (0.125 sec)

COMPLEMENTO

La diferencia simétrica es una operación que devuelve los elementos que están en uno de los dos conjuntos de datos, pero no en ambos conjuntos de datos. Matemáticamente, la diferencia simétrica entre dos conjuntos A y B se denota como $A \Delta B$. Se puede definir como:

$$A \Delta B = (A - B) \cup (B - A)$$



Donde “-” denota la diferencia entre conjuntos y “U” denota la unión de conjuntos. En otras palabras, la diferencia simétrica de dos conjuntos es el conjunto de elementos que están en A o en B pero no en ambas.

→ Comprehension

$$A = \{\text{idusuario, nombre, apellido, cedula, idcargo, idrol}\}$$

$$B = \{\text{idcargo, descripción}\}$$

$$A' = \{\text{descripción}\}$$

→ Extension

$$A = \{x | x = (\text{idusuario, nombre, apellido, cedula, idcargo, idrol})\}$$

$$B = \{x | x = (\text{idcargo, descripción})\}$$

$$B/A = \{x | x \in B \text{ y } x \notin A\}$$

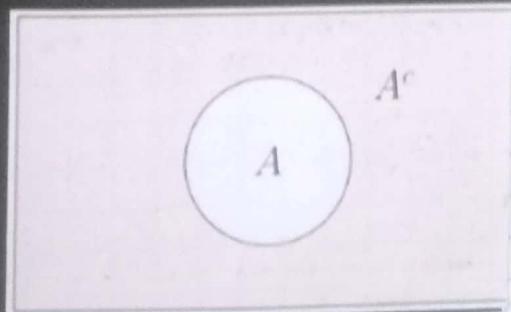
$$B/A = \{x | x \text{ pertenece a } B \text{ y } x \text{ no pertenece a } A\}$$

MariaDB [factcom]> Select cargo.descripcion

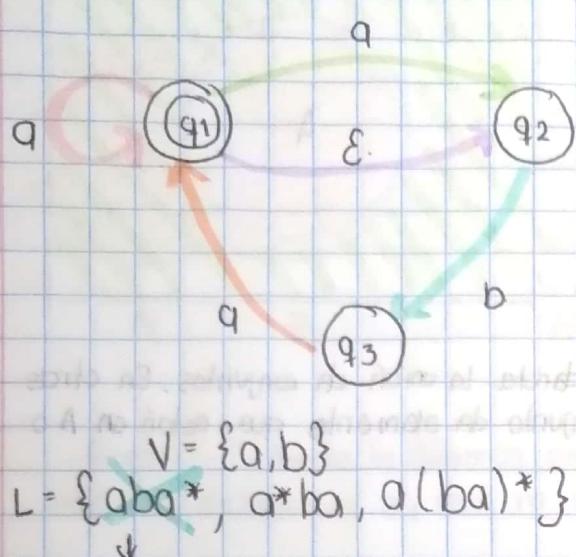
-> From cargo

-> where cargo.idcargo NOT IN (Select ususario.idcargo From ususario);

```
+-----+
| descripcion |
+-----+
| operativo   |
| auxiliar    |
+-----+
2 rows in set (0.001 sec)
```



Se genera una palabra vacía cuando el estado inicial y el final son los mismos



No puede haber una sin a.

$$a^* (aba^* \mid ba) a^* \quad l=0$$

$$(a^* ba^*)^*$$

↓
no va

$$(a^* ba)^*$$

- λ
- aba
- aabaabaa

Cadenas aceptadas

λ
aba
abbaaba
ba
aba
aabg
adabg
a
aba
ababa

ANTECEDENTES

LENGUAJE: Conjunto de signos y reglas que permiten la comunicación con un ordenador.

GRAMÁTICA: Ciencia que estudia los elementos de una lengua y sus convenciones.

AUTÓMATAS: (maquinas abstractas).

GRAMÁTICA, LENGUAJES Y LOS AUTÓMATAS QUE LOS RECONOCEN:

→ TIPO 0 (SIN RESTRICCIONES)

- Recorridos no enumerables.
- Máquina de Turing → Determinista / No determinista (la cinta)

→ TIPO 1 (SENSIBLE AL CONTEXTO)

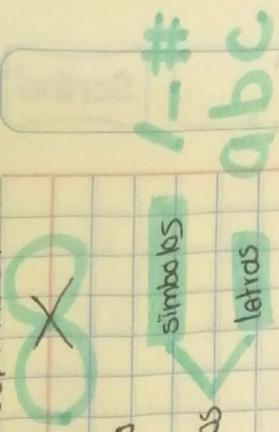
- Máquina de Turing

→ TIPO 2 (LIBRE DE CONTEXTO) INDEPENDIENTES DEL CONTEXTO (LC)

- Autómata de Pila
- Determinista (APP)
- No Determinista (APN)

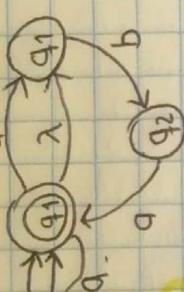
→ TIPO 3 (REGULARES)

- Autómata Finito
- Determinista (AFD)
- No Determinista (APN)



- Conjunto finito
- No vacío.
- Da elementos
- lertos

ALFABETO



EJEMPLO

DIAGRAMAS
DE TRANSICIÓN

Estado inicial

Estado final

El estado inicial
TAMBIEN
puede ser el estado
final

Marcan el
sentido

TEORÍA DE LOS

LENGUAJES

Patrones

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

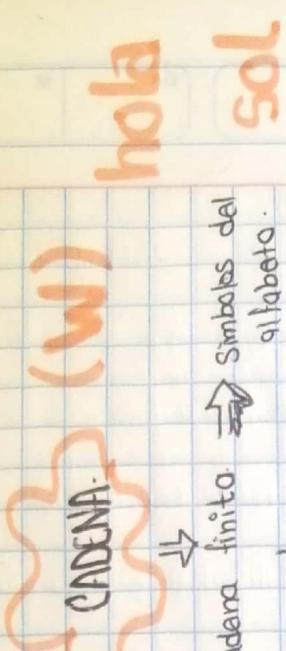
 LENGUAJES
FORMALES

Se definen mediante
una propiedad.

EXTENCIÓN

Son finitos. → se definen todos
los elementos

Ejemplo

 $L = \{a, i, f, else\}$ Se enumeran
entre llaves

cadena finita → Símbolos del alfabeto.

LONGITUD
(|w|)
cadena finita → Número de letras
Número de símbolos.

$$\begin{array}{r} \text{com} \\ -1 \quad 2 \quad 3 \quad 4 \quad 5 \end{array}$$

Longitud 0 = Palabra vacía

$$\begin{array}{r} V = \{a, b\} \\ W_1 = bba \\ W_2 = bbabbba \end{array}$$

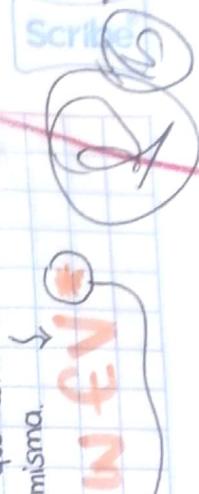
$\forall n \rightarrow$ conjunto de todas las palabras de longitud n

POTENCIA DE UNA PALABRA

Operación que concatenan la palabra n veces
consigo misma.

ESTRELLA
DE UN

Indica que se repite.
repetición de 0 a n veces



Basados en
el alfabeto
→ Comunicar.

Lenguaje → Conjunto

EXTENCIÓN

se definen todos
los elementos

Ejemplo

UNIDAD 2

D M A

Scribe

EXPRESIONES REGULARES

Una cadena aceptada se construye a través del paso por las flechas que se dibujan desde un estado inicial a un estado de aceptación.

- Sobre cada flecha se puede colocar un símbolo, dos símbolos o un conjunto de símbolos
- Un lenguaje está formado por cadenas aceptadas



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

$$\begin{aligned} \text{letra} &= \{[a-z], [A-Z]\} \rightarrow \text{CONJUNTOS} \\ \text{digito} &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \end{aligned}$$

No es válido
No llega al estado de aceptación

$$L = \{a, aa, abc, a1, a1a22, a444\} \rightarrow \text{Siempre comienza con letras.}$$

$$\begin{aligned} L &= \{w \in \{\text{letra, digito}\}^* \mid w \text{ cumple con la propiedad de } \text{letra} (\text{letra}^* \mid \text{digito}^*)^*\} \\ L &= \{\text{letra} (\text{letra}^* \mid \text{digito}^*)^*\} \end{aligned}$$

→ Si no se coloca un estado final en diagrama no es NFA / o sin estado inicial

fórmula que me indica cuáles son las palabras aceptadas → EXPRESIÓN REGULAR

→ DEFINICIÓN FORMAL DE EXPRESIÓN REGULAR.

Una EXPRESIÓN REGULAR (para un alfabeto Σ) se define formalmente como sigue:

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

Los lenguajes regulares son aquellos que son reconocidos por automatas finitos y son también aquellos lenguajes generados por gramáticas regulares

u
n
i
g
a
d
a
2

UNIDAD 2

D

M

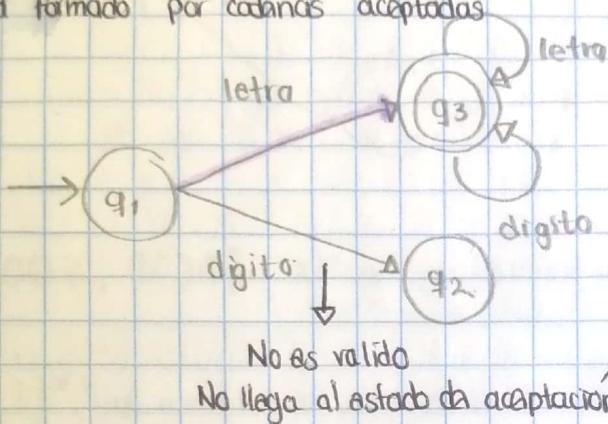
A

Scribe®

EXPRESIONES REGULARES

Una cadena aceptada se construye a través del paso por los flechas que se dibujan desde un estado inicial a un estado de aceptación.

- Sobre cada flecha se puede colocar un símbolo, dos símbolos o un conjunto de símbolos.
- Un lenguaje está formado por cadenas aceptadas.



$$V = \{\text{letra, dígito}\}$$

$$\begin{aligned} \text{letra} &= \{[a-z], [A-Z]\} \rightarrow \text{conjuntos} \\ \text{dígito} &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \end{aligned}$$

$$L = \{a, aa, abc, a1, a1a22, a444\} \rightarrow \text{Siempre comienza con letras.}$$

$$L = \{w \in \{\text{letra, dígito}\}^* \mid w \text{ cumple con la propiedad de } \text{letra} (\text{letra}^* \mid \text{dígito}^*)^*\}$$

$$\downarrow \\ L = \{\text{letra} (\text{letra}^* \mid \text{dígito}^*)^*\}$$

→ Si no se coloca un estado final es diagrama no SFA / o sin estado inicial

fórmula que me indica cuáles son las palabras aceptadas → EXPRESIÓN REGULAR

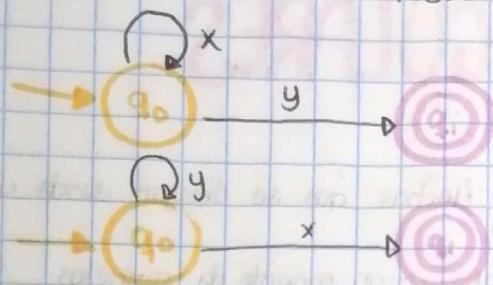
→ DEFINICIÓN FORMAL DE EXPRESIÓN REGULAR.

Una EXPRESIÓN REGULAR (para un alfabeto Σ) se define formalmente como sigue:

- \emptyset es una expresión regular.
- Cada miembro de Σ es una expresión regular.
- Si p y q son expresiones regulares, también lo es $(p \cup q)$.
- Si p y q son expresiones regulares tambien $(p \cdot q)$ → p concatenada con q .
- Si p es una expresión regular tambien lo es p^* .

Los lenguajes regulares son aquellos que son reconocidos por automatas finitos y son también aquellos lenguajes generados por gramáticas regulares.

EJEMPLOS DE EXPRESIONES REGULARES



Acepta al lenguaje que consiste en una o más x seguidas por una sola y .

Acepta al lenguaje que consiste en una o más x seguidas por una sola y .

$L = \{x^k y x^l \mid k \geq 0, l \geq 0\}$

• $\Sigma = \{x, y\}$ lenguaje de entrada

• $M = \{q_0, q_1\}$

• $\delta(q_0, x) = q_0$

• $\delta(q_0, y) = q_1$

• $\delta(q_1, x) = q_1$

• $q_f = q_1$ (último x de la cadena)

• $L = \{x^k y x^l \mid k \geq 0, l \geq 0\}$

• $\Sigma = \{x, y\}$ lenguaje de entrada

• $M = \{q_0, q_1\}$ estados

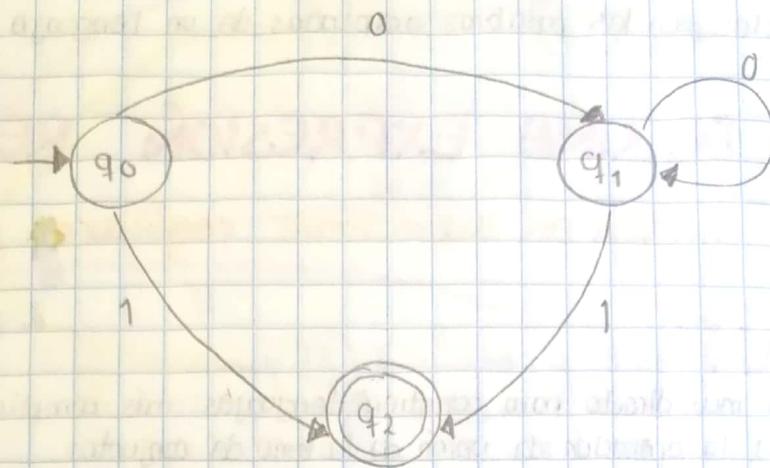
• $\delta(q_0, x) = q_0$

• $\delta(q_0, y) = q_1$

• $\delta(q_1, x) = q_1$

• $\delta(q_1, y) = q_0$

• $q_f = q_0$ (último x de la cadena)



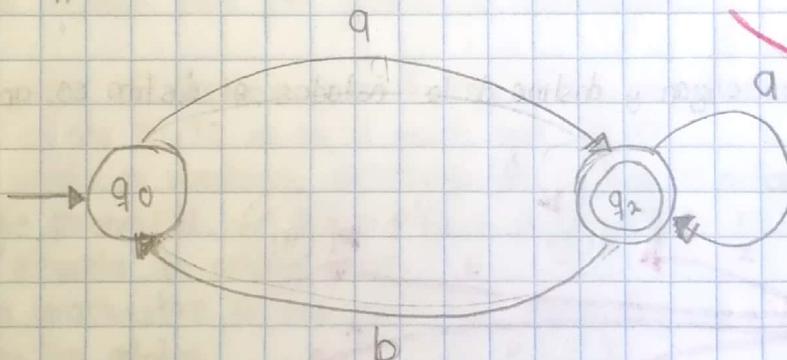
* → 0 a más veces.
+ → 1 o más veces.

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

$L = \{1, 01, 001, 0001\} \rightarrow$ Tiene que terminar en 1
El 0 pueda estar 0 a N veces.

$L = \{w \in \{0, 1\}^* \mid w \text{ cumple la propiedad de } 1 | 0^+ 1\}$

$$\underline{\underline{L = \{1 | 0^+ 1\}}} +$$



$$V = \{a, b\}$$

$L = \{a, a\underline{a}, a\underline{aa}, ab\underline{a}, aab\underline{a}\} \rightarrow$ Tiene que comenzar con a
Tiene que terminar en a
a y b pueden o no aparecer.

$L = \{w \in \{a, b\}^* \mid w \text{ cumple la propiedad de } a^+ (ba^*)^*\}$

$$a^+ (ba^*)^0 = a / aaaa$$

$$a^+ (ba^*)^1 = aba, aaba,$$

$$a^+ (ba^*)^2 = abaa / aa baa$$

$$a^+ (ba^*)^3 = ababaaaa$$

EXPRESIÓN REGULAR

Forma que indica cuáles son las palabras aceptadas de un lenguaje.

2.2 DISEÑO DE UNA EXPRESIÓN REGULAR.

Operaciones que existen en expresiones regulares

1 UNIÓN (U)

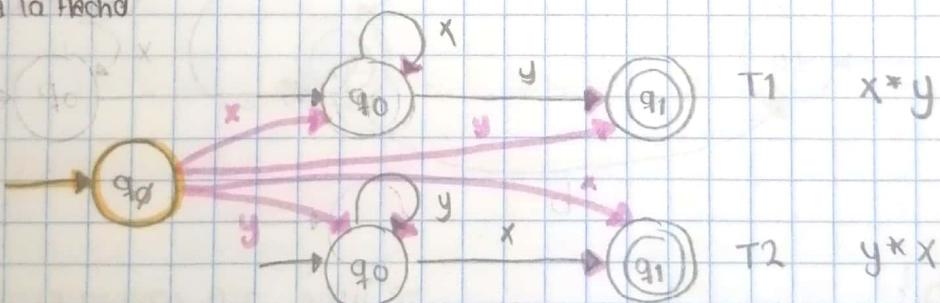
Representa la técnica más directa para construir lenguajes más complicados a partir de los más básicos utilizando la operación de unión de la teoría de conjuntos.

→ PASOS

1. Dibuja un nuevo estado inicial
2. Dado el como un estado de aceptación si y solo si uno de los estados iniciales anteriores era de aceptación.
3. Para cada estado que sea punto de destino de una flecha de alguno 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 si es que aún existen.

NOTA: Las flechas son origen y destino de los estados, al destino se apunta con la punta de la flecha

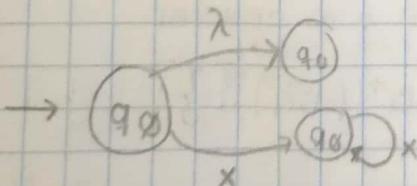
PASO 1



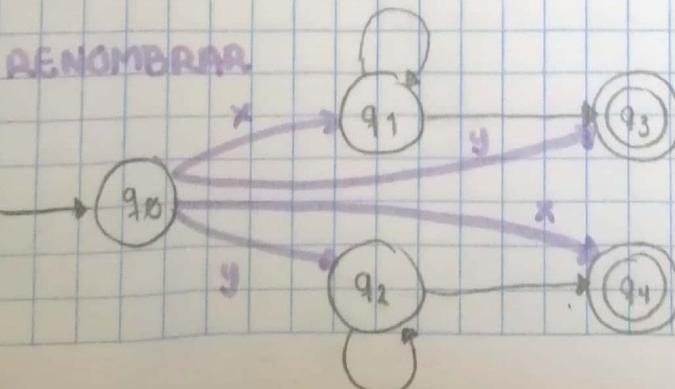
PASO 2

PASO 3

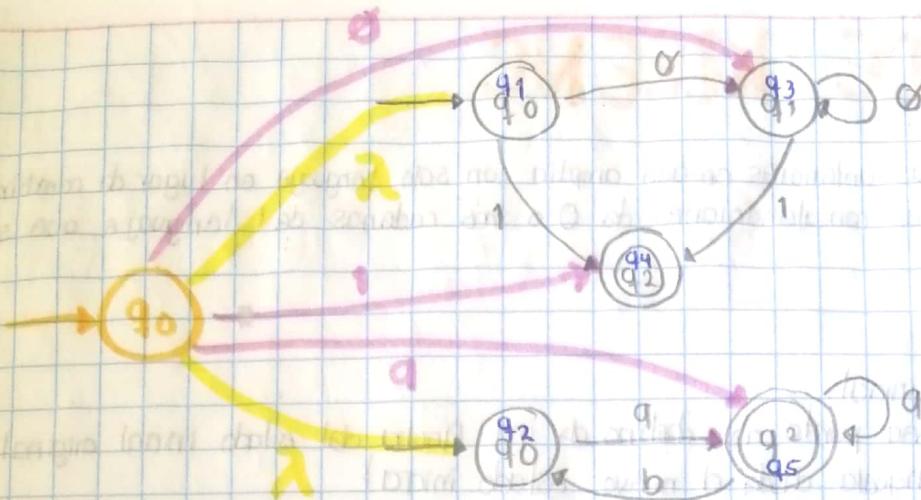
PASO 4



PASO 5 - RENOMBRAR.



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



PASO 1

PASO 3

PASO 4

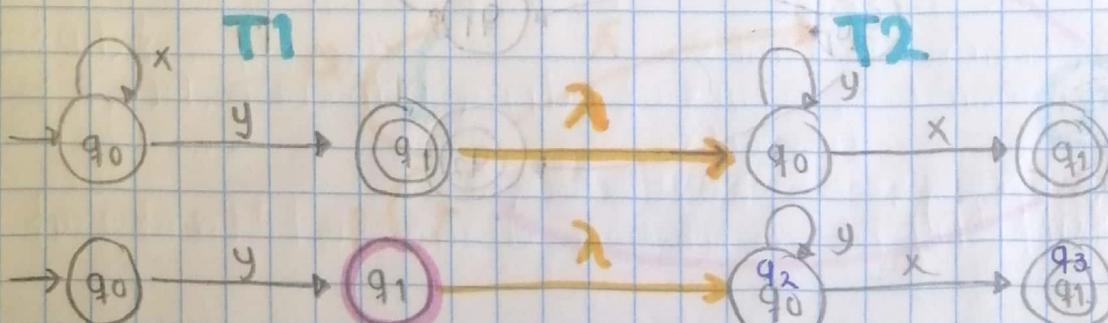
PASO 5

$$L = \{ (1|0^+1) \cup a^+(ba^+)^* \}$$

2 CONCATENACIÓN

Es una técnica para recopilar todos las cadenas formadas al concatenar una cadena de primer lenguaje y una cadena del segundo. A la colección de cadenas formadas de esta manera se denomina concatenación de dos lenguajes. Pasos para generar la concatenación:

1. Colocar en orden los diagramas, uno enfrente del otro, dependiendo del lenguaje que se desea obtener, ya que $T_1 \cdot T_2$ no es lo mismo que $T_2 \cdot T_1$
2. Unir los estados de aceptación del diagrama T_1 con el estado inicial de T_2 , o bien si este colorado primero T_2 , unir los estados de aceptación de T_2 con el inicial de T_1 y colocar la etiqueta de palabra vacía.
3. Borrar la característica de aceptación del primer diagrama
4. Renombrar los estados.



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

PASO 1

PASO 2

PASO 3

PASO 4 RENOMBRAR.

T1 · T2



ESTRELLA DE KLEENE

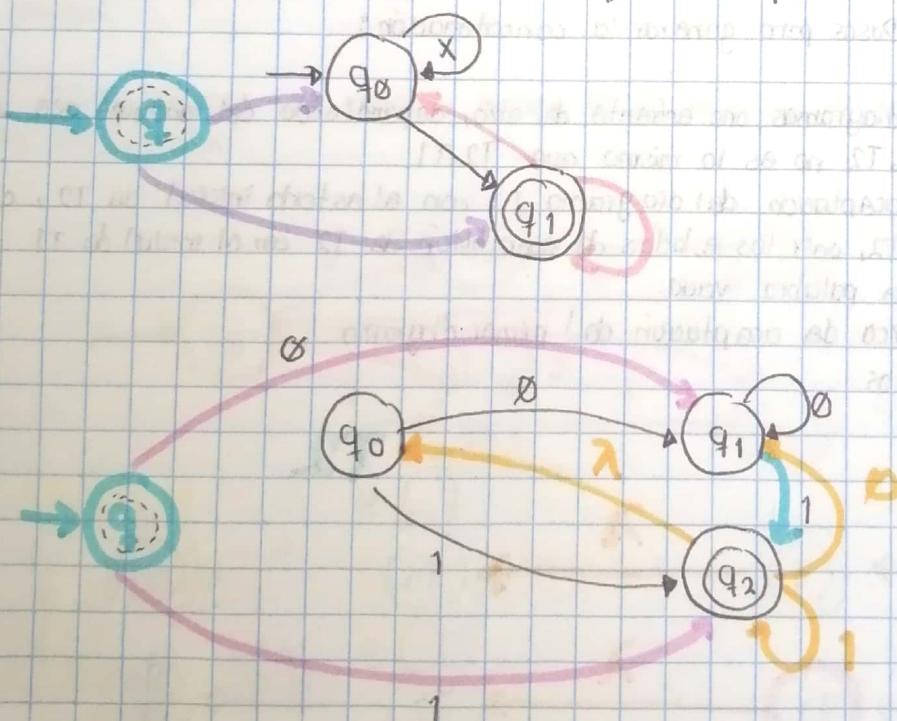
Esta operación difiere de las anteriores en que amplia un solo lenguaje en lugar de combinar dos. Esto se logra formando todas las concatenaciones de 0 o más cadenas del lenguaje que se amplía.

PRIMERA ETAPA:

- Dibujar un nuevo estado inicial.
- Por cada estado que sea punto de destino de una flecha del estado inicial original, dibujamos una flecha con la misma etiqueta dada el nuevo estado inicial.
- Designar a este nuevo estado inicial como estado de aceptación.
- Verificar que no existan más de un estado inicial como en el diagrama final, en caso de existir concatenar al nuevo estado inicial con los estados originales y colocar la etiqueta correspondiente.

SEGUNDA ETAPA

- Añadir una flecha de cada estado de aceptación a cada estado que es destino del inicial.
- Rotular estos nuevos arcos con la etiqueta correspondiente.



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

	0	1	λ
q	q_1	q_2	q_0
$\rightarrow q_0$	q_1	q_2	
$F q_2$	q_1	q_2	q_0

APLICACIONES DE LAS ENTIDADES

D M A
02 Octubre 2024



- a) En la construcción de compiladores, con el fin de reconocer cuales son las cadenas de simbolos del programa fuente que deben considerarse como representaciones de objetos individuales, como las constantes numéricas, los nombres de variables y las palabras reservadas.
- b) Inteligencia artificial: En la creación de máquinas de estado finito con el fin de reconocer la gramática y los lenguajes formales para la aceptación de simbolos y reconocimiento de cadenas que realiza un agente capaz de actuar o interpretar instrucciones específicas.
- c) En electronica: Las máquinas de estado finito son aceptadoras de simbolos y en electronica se emplean en la creación de circuitos ya que se tiene que reconocer que una cadena de simbolos sea idéntica a otra y así dejar pasar los datos, ejemplos de ellos son: los semáforos, los apagadores automáticos de luz, los sistemas automáticos de trago entre otros.

→ Comienzo apuntes unidad 1 y 2 → rendir en junio