Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
Primer Semestre 2020
Ing. Mario Bautista, Ing. Manuel Castillo, Ing. Kevin Lajpop
Aux: Luis Mazariegos, Huriel Gómez, Fernando Paz, Juan Landaverde

Práctica 1

Objetivos

General:

1. El estudiante aplique los conceptos sobre la fase de análisis léxico de un compilador en una solución de software.

Específico:

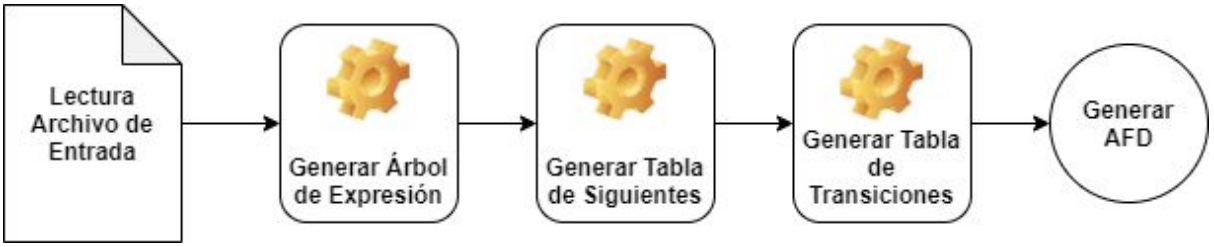
1. El estudiante refuerce los conocimientos del método del árbol, expresiones regulares y autómatas finitos deterministas.
2. Identificar y programar el proceso de reconocimiento de lexemas mediante el uso de autómatas finitos deterministas.

Descripción

Se solicita a usted como estudiante que desarrolle una aplicación con el principal fin de analizar expresiones regulares generando a su salida un autómata finito determinista con la validación de lexemas.

Su funcionalidad principal será el interpretar expresiones regulares permitidas, por medio del análisis de un archivo de entrada, el cual contendrá expresiones regulares permitidas por el lenguaje.

Flujo del Programa



- Se deben poder mostrar todas las gráficas de los árboles generados por archivo en una galería dentro de la aplicación.
- También deben mostrarse la tabla de siguientes, tabla de transiciones y del AFD.

Interpretación de Expresiones Regulares

Este realiza la lectura de un archivo de entrada (.er) el cual contiene una o más expresiones regulares que serán reconocidas en el lenguaje.

Definición de Lenguaje

Comentarios
Será permitido el manejo de comentarios de una o más líneas, pueden ser de 2 formas:

- **Comentario de una sola línea:**

```
// Este es un comentario
```

- **Comentarios multilínea**:

```
<!
    Este es un comentario
    multilínea
!>
```

Estructura del archivo
El archivo que contendrá las expresiones regulares será compuesto de una estructura en la cual se encontrarán un identificador seguido su respectiva expresión regular, seguido un doble porcentaje. A continuación se encontrará uno o más identificadores seguido de un lexema de entrada.

```
{
ID1 -> Expresion_Regular ;
ID2 -> Expresion_Regular ;
ID3 -> Expresion_Regular ;
//Mas sentencias…..

%%
%%

IDA: "Lexema de entrada" ;
IDB: "Lexema de entrada" ;
IDC: "Lexema de entrada" ;
//Mas sentencias
}
```

- Cada sentencia será delimitada por un punto y coma (**;**).

Reconocimiento de Expresiones Regulares

Para el análisis y evaluación de cada una de las expresiones regulares se deberán utilizar una notación prefija.

Descripción de la notación a utilizar:

Notación	Descripción
. a b	Concatenación entre a y b.
| a b	Disyunción entre a y b.
? a	a, 0 o una sola vez.
*** a**	a, 0 o más veces.
+ a	a, 1 o más veces.

Macros o Conjuntos

Un conjunto o macro, son agrupaciones de caracteres del mismo tipo, permitidos en el lenguaje, como agrupaciones de letras, números, etc.
La palabra reservada a utilizar será “CONJ”.

Notación	Descripción
f~j	Conjunto de letras {f,g,h,i,j}.

a~z	Conjunto de letras minúsculas de la ‘a’ a la ‘z’.
A~Z	Conjunto de letras minúsculas de la ‘A’ a la ‘Z’.
0~25	Conjunto de números de 0 a 25.
1,3,5,11,13	Conjunto de números {1,3,5,11,13}.
n,N,e,E	Conjunto de letras {n,N,e,E}.
!~&	Conjunto de signos desde ‘!’ hasta ‘&’ (33 al 38 en código ascii). **Nota:** El rango válido será desde el ascii 32 hasta el 125 omitiendo los ascii de las letras y dígitos.

Notas:

- Un conjunto puede utilizarse dentro de una expresión regular.
- Un conjunto no puede utilizarse en la definición de otro conjunto.
- El uso de conjuntos dentro de las expresiones regulares serán delimitados por corchetes { }.

Ejemplo:

En este caso se definen los conjuntos ‘letra’ y ‘digito’ y posteriormente se utilizan en las expresiones regulares

.

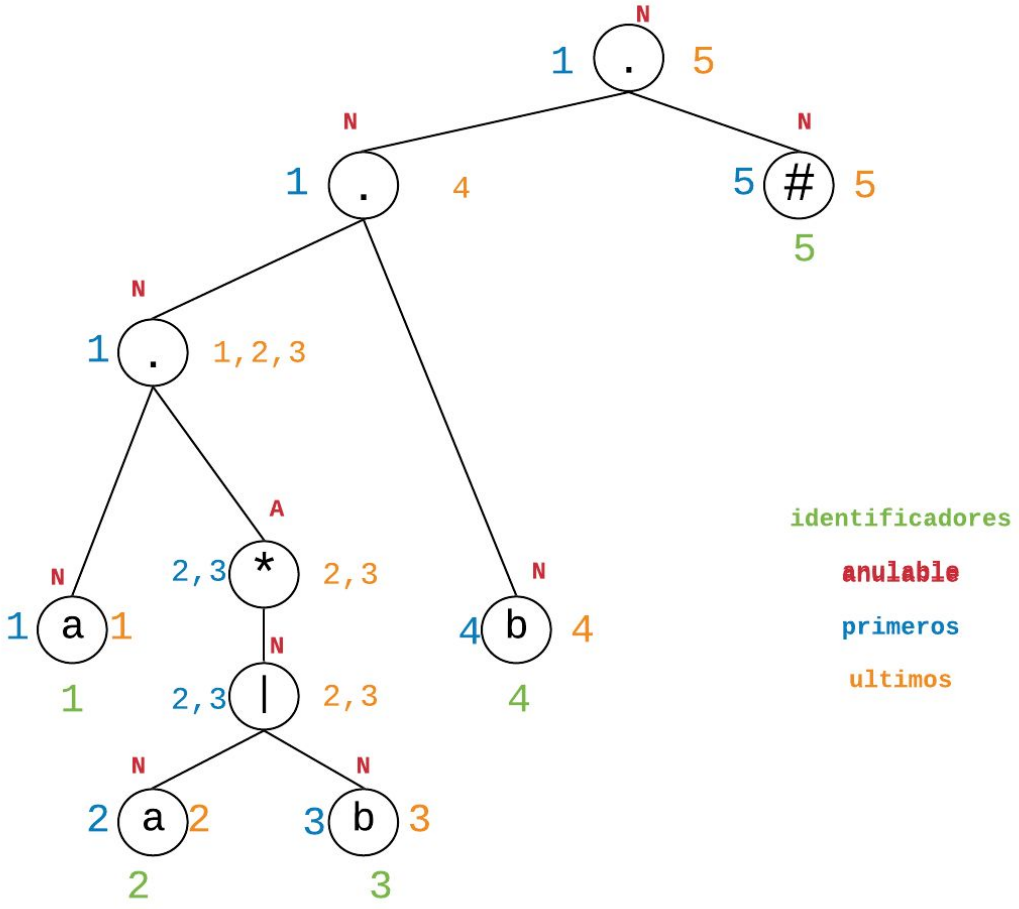
```
{
// Conjuntos
CONJ: letra -> a~z;
CONJ: digito -> 0~9;

<! Expresiones
   Regulares
!>
ExpReg1 -> . {letra} * | "_" | {letra} {digito};
RegEx2 -> . {digito} * | "_" | {letra} {digito};
ExpReg3 -> . {digito} . "." + {digito};

%%
%%

ExpReg1 : “este_es_un_lexema_valido”
ExpReg3 : “34.44”

}
```

Generación de AFD

Árbol de expresión

Cada expresión regular descrita con anterioridad en el archivo de entrada deberá generar un árbol de expresión igual al que se muestra a continuación. Cada hoja del árbol tendrá los siguientes atributos dentro de la gráfica.

- Identificador
- Anulable | No Anulable
- Primeros
- Ultimos



(Gráfica del Árbol de expresión)

Tabla de transición y tabla de siguientes

Con el árbol de expresión generado con anterioridad se deberá generar una tabla de transición y una tabla de siguientes correspondientes.

Hoja		Siguientes
A	1	2,3,4
A	2	2,3,4
B	3	2,3,4
B	4	5
#	5	--

- **Tabla de siguientes**

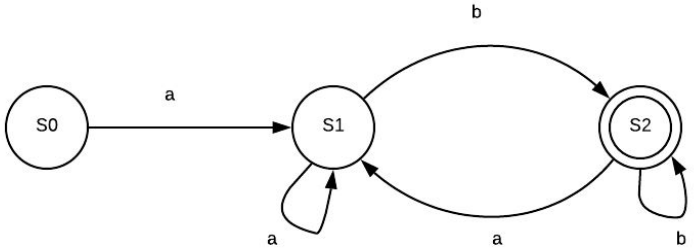
Estado	Terminales	
	a	**b**
S0 {1}	S1	--
S1 {2,3,4}	S1	S2
S2 {2,3,4,5}	S1	S2

- **Tabla de transiciones**

  Estado de aceptación.

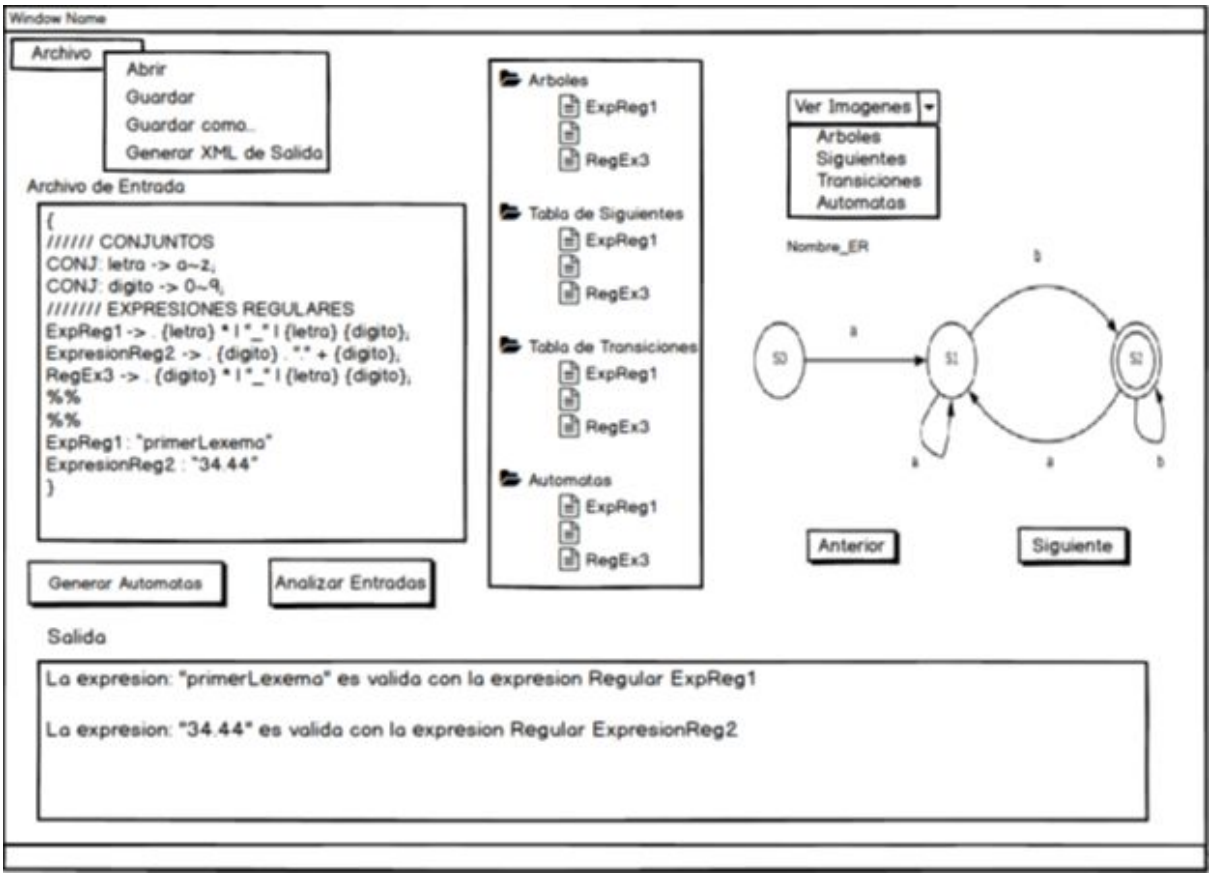
Autómata finito determinístico

Cada tabla de transición deberá usarse para generar el autómata finito determinístico siguiente:



Interfaz Sugerida

Consola: la consola deberá mostrar el siguiente mensaje: La expresión: **<lexema de entrada>** es válida con la expresión Regular **<Nombre de la Expresión Regular**



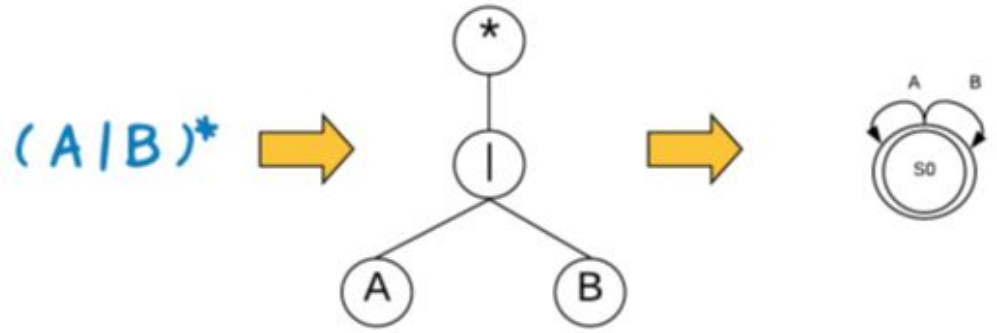
Anexos

Explicación sobre Método del Árbol

Sirve para encontrar un AFD mínimo dada una expresión regular. Las características de un AFD son:

- No tiene transiciones Epsilon
- No tiene dos o más transiciones con el mismo símbolo a un mismo estado

En el ejemplo se puede observar la entrada, y la salida esperada del método del árbol.

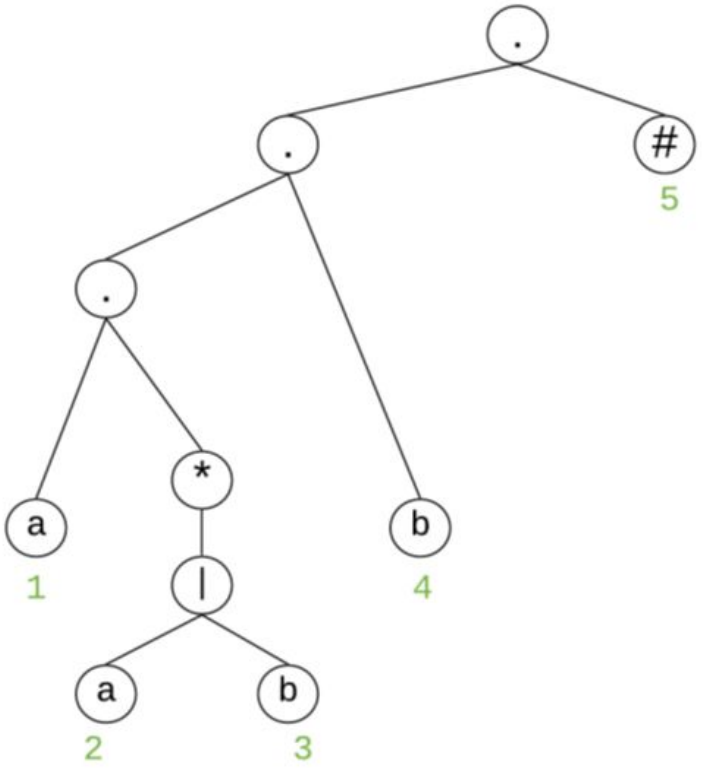


Pasos a seguir para realizar el método del árbol

- *Agregar* al final de la expresión regular el símbolo #

$$a(a|b)^*b \rightarrow a(a|b)^*b\#$$

- *Construir* el árbol de Sintaxis y dar un identificador único a cada hoja

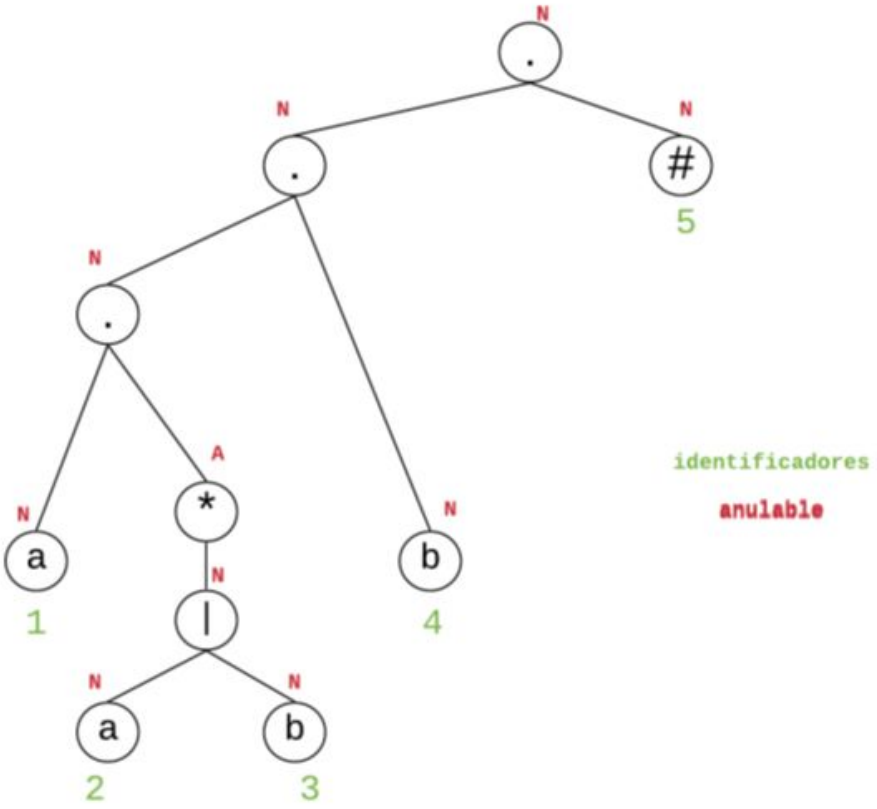


- Determinar los nodos anulables

Se debe basar utilizando las siguientes reglas:

Terminal	No anulable
C1*	Anulable
C1+	No Anulable (Si C_1 es no anulable)
C1?	Anulable
C1|C2	(Anulable(C_1)||Anulable(C_2))?Anulable:No Anulable
C1C2	(Anulable(C_1)&&Anulable(C_2))?Anulable:No Anulable

La salida esperada, utilizando las reglas anteriores es:

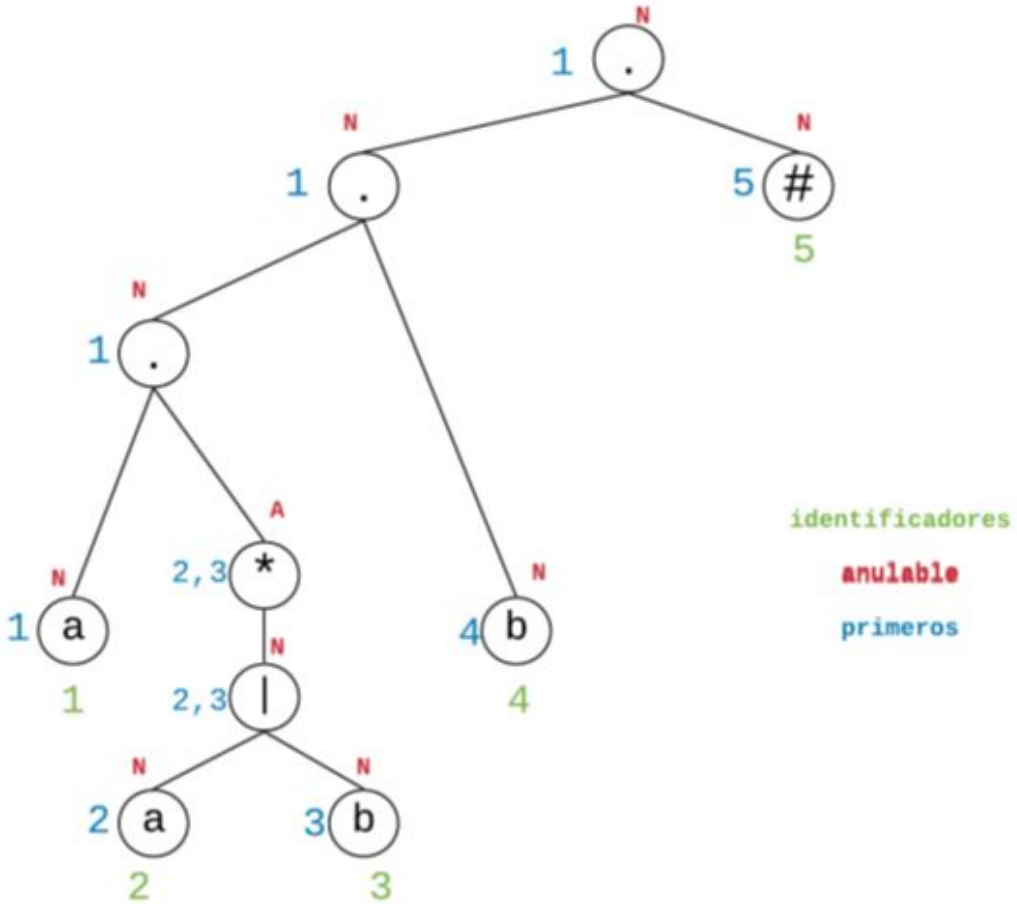


- Determinar los nodos primeros

Se debe basar utilizando las siguientes reglas:

terminal	terminal
C1*	Primeros(C1)
C1+	Primeros(C1)
C1?	Primeros(C1)
C1|C2	Primeros(C_1) + Primeros(C_2)
C1C2	(Anulable(C_1))?(Primeros(C_1) + Primeros(C_2)):Primeros(C_1)

La salida esperada, utilizando las reglas anteriores es:

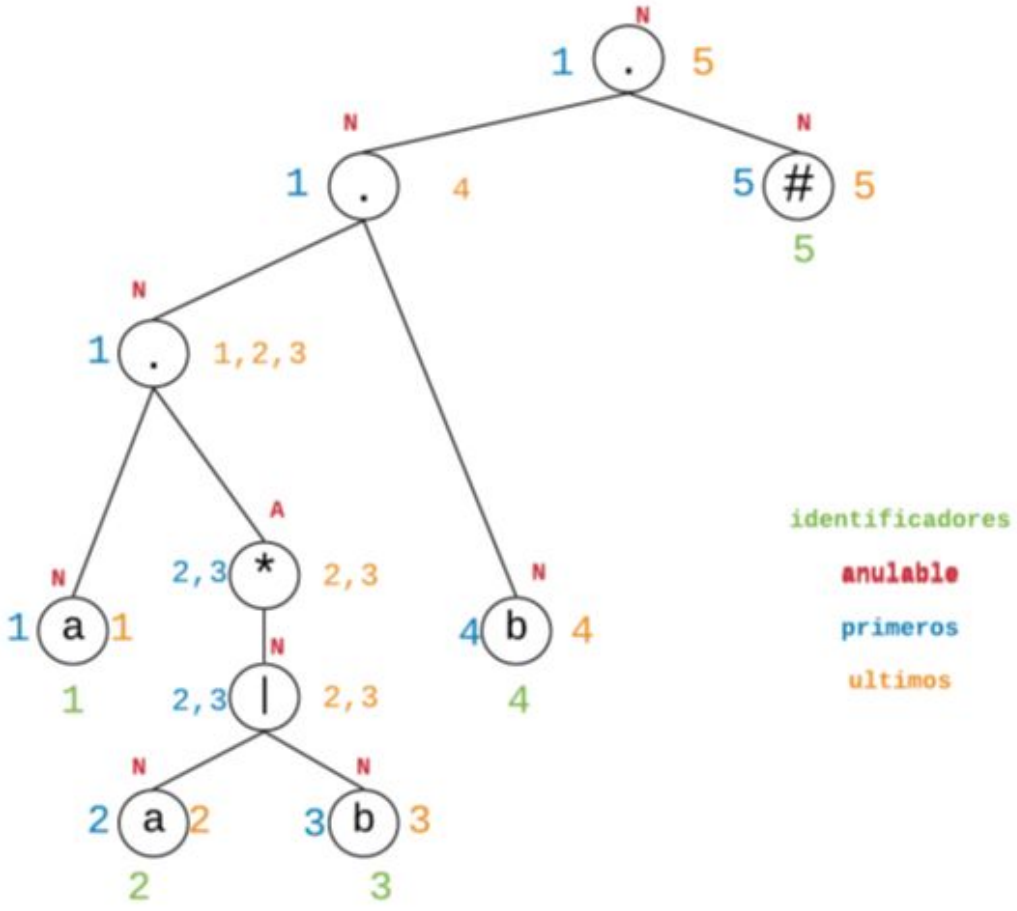


- *Determinar los nodos últimos*

Se debe basar utilizando las siguientes reglas:

terminal	terminal
C1*	ultimo(C_1)
C1+	ultimo(C_1)
C1?	ultimo(C_1)
C1|C2	ultimo(C_1) + ultimo(C_2)
C1C2	(anulable(C_2))?(ultimo(C_1) + ultimo(C_2)):ultimo(C_2)

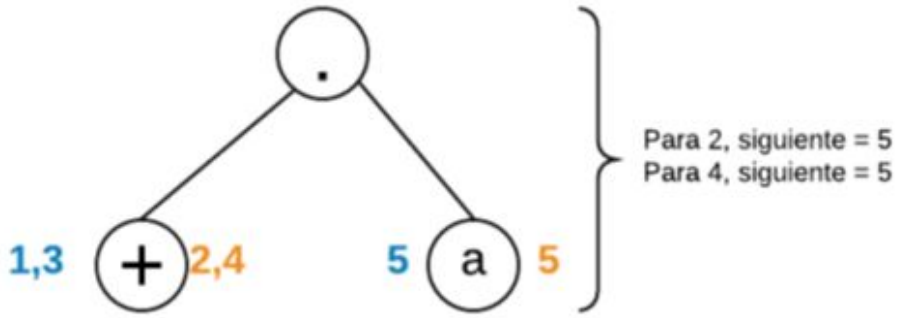
La salida esperada, utilizando las reglas anteriores es:



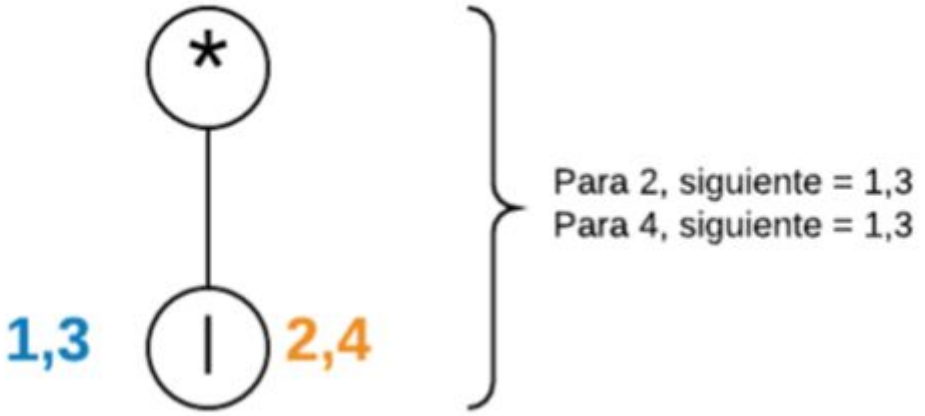
- Crear Tabla de Siguientes

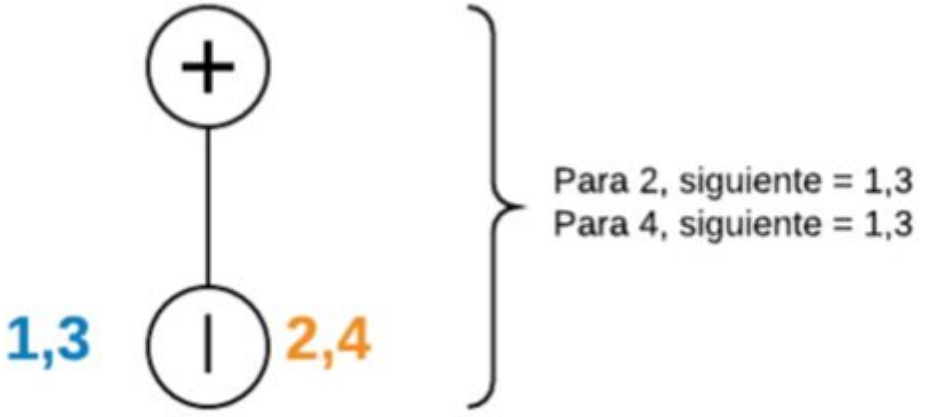
Solamente Concatenación (·), Cero o Muchas veces (*) y Una o muchas veces (+) tienen Siguiente

En la concatenación los Siguientes para los últimos de C_1 son los primeros del Nodo C_2



Para * y + los siguientes para los últimos de C_1 son los primeros de C_1





Hoja		Siguientes
a	1	2,3,4
a	2	2,3,4
b	3	2,3,4
b	4	5
#	5	--

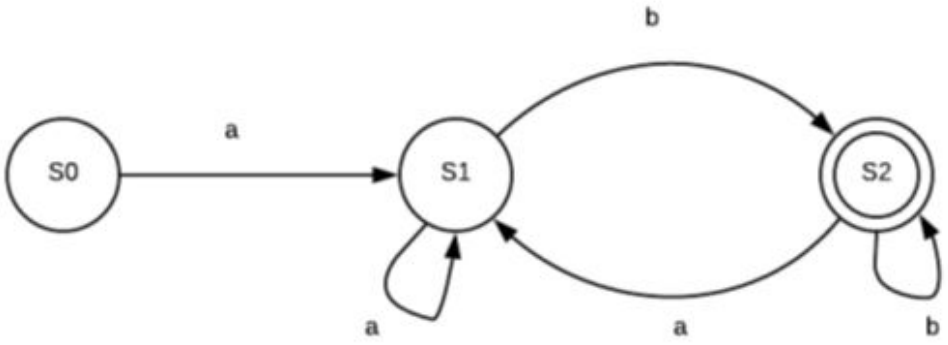
- Construir la tabla de Transiciones

Estado	**Terminales**	
	a	**b**
S0 {1}	S1	--
S1 {2,3,4}	S1	S`2
S2 {2,3,4,5}	S1	S2

El estado inicial son los identificadores que estén en los primeros del nodo raíz

Un estado de aceptación es todo aquel que tenga el identificador del símbolo # agregado inicialmente.

- *Ilustración del AFD construido utilizando la tabla de Transiciones*



Entregables

---

1. Código fuente de la solución.
2. Ejecutable.
3. Manual Técnico y de Usuario.
4. Link del repositorio de versiones.

Consideraciones

---

1. Trabajar de manera individual
2. No es permitido el uso de herramientas para el análisis. Deben de construir el analizador por medio del metodo del arbol.
3. Lenguaje de programación: Java.
4. El sistema operativo y el IDE son libres.
5. Cualquier copia total o parcial será reportada y no permitirá la continuidad en el laboratorio.
6. No se calificará desde el IDE.
7. Los archivos de entrada serán proporcionados el día de la calificación.
8. No habrá prórroga y entregas fuera de horario tienen una nota de CERO.
9. Fecha de entrega **viernes 21 de febrero a las 23:59.**