

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Integrantes:

Adrián Aguilera Moreno

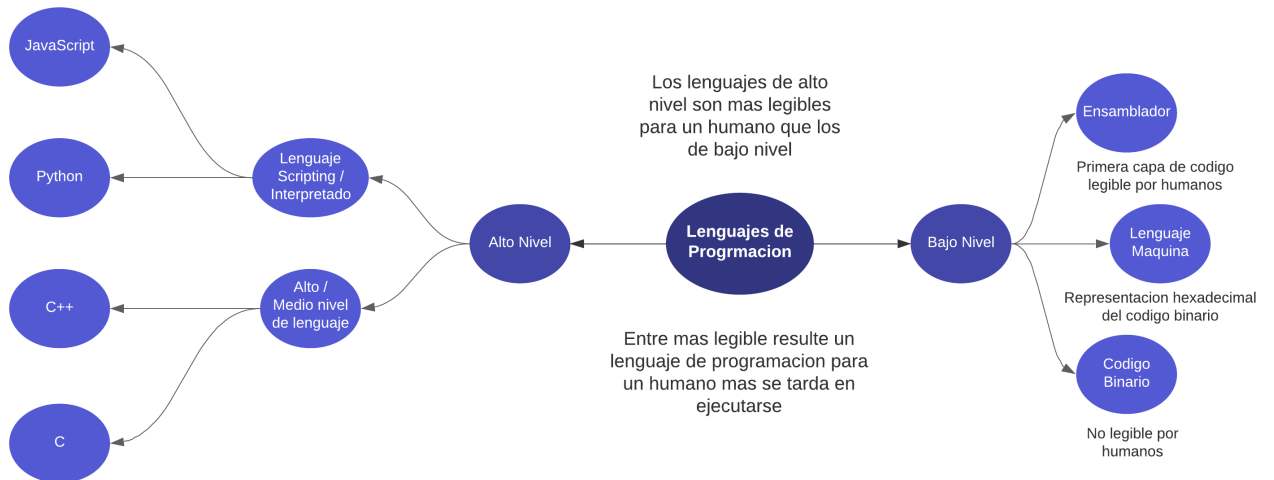
Sebastián Alejandro Gutierrez Medina



Compiladores

Tarea 01

Pregunta 1.



Pregunta 2.

La siguiente tabla define los tokens para un lenguaje simple donde $\Sigma = \{a, \dots, z, 0, \dots, 9, \oplus, (,)\}$

<i>token</i>	<i>exp. regular</i>
num	$0 + [1 - 9][0 - 9]^*$
lam	"lam"
dot	.
lp	(
rp)
binop	\oplus

- Extiende la tabla anterior para agregar un token para identificadores donde la primera letra debe ser mayúscula seguida de cualquier secuencia de letras o números.
- Construye un autómata finito determinista que acepte los tokens descritos en la tabla. Puedes usar algún método, eg. derivadas de expresiones regulares o construcción de un AFN_ϵ y transformaciones. Indica el método usado y muestra el proceso.

Solución.

- Extendiendo la tabla anterior tenemos que

<i>token</i>	<i>exp. regular</i>
num	$0 + [1 - 9][0 - 9]^*$
lam	"lam"
dot	.
lp	(
rp)
binop	\oplus
mi	$[A-Z][a-zA-Z0 - 9]^*$

- Para diseñar este autómata utilizaremos el método de derivaciones en expresión regular. Así, nuestra expresión regular sería

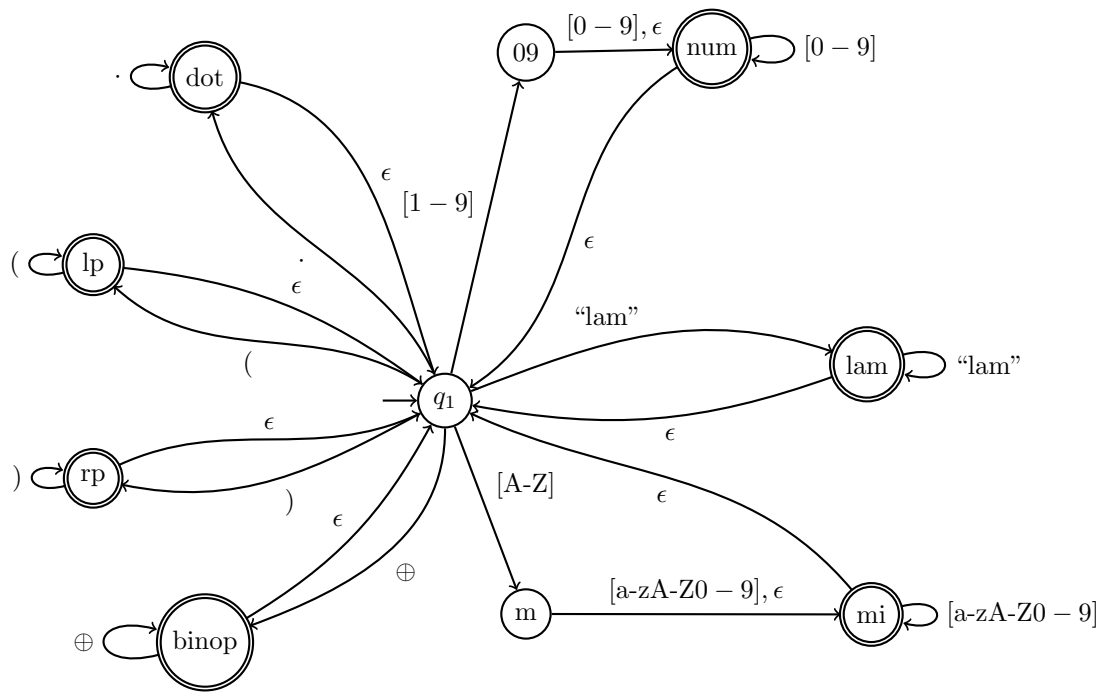
$$\text{token} = (\text{num} \mid \text{lam} \mid \text{dot} \mid \text{lp} \mid \text{rp} \mid \text{binop} \mid \text{mi})$$

nuestro autómata estará definido por la expresión

$$\text{token token}^*$$

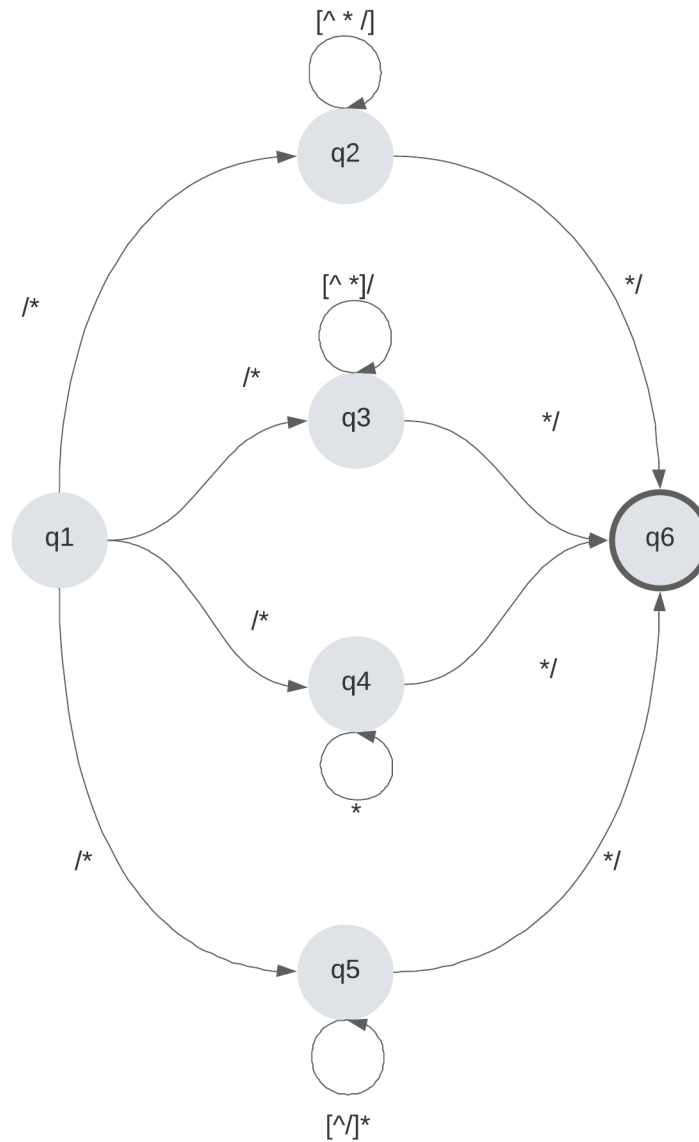
Obs. No aceptamos a la cadena vacía, por tanto nuestro autómata debe tener un estado inicial que no sea terminal y del cuál sus transiciones a otros estados sean no vacías. Cada estado final debe poder decidir si termina o regresa al inicio de nuestro autómata (pues el token en cuestión puede formarse de los diferentes tokens en la tabla).

A continuación se da la gráfica que representa el autómata AFN_{ϵ} solicitado, este es



Pregunta 3.

- Dado que un comentario valido en c seria `/**/` y en base a la expresión regular brindada esta cadena no es aceptada, no es posible que la expresión regular `/*([^*/][^*/|/^/])**/`
- Autómata que acepta comentarios de c :

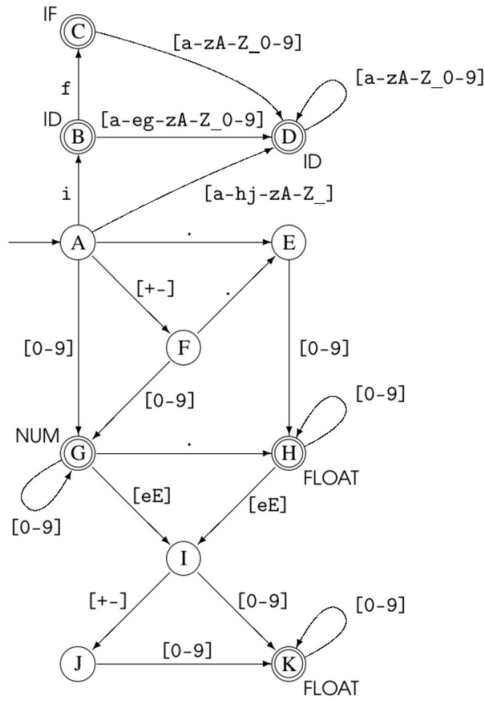


Expresión regular que corresponde a los comentarios en c

`/* ([^*/] ([^*/|/])* | ([^/])* | [^*/]+)* [*/]* */`

Pregunta 4.

Considera el siguiente autómata



- a) ¿Cuál es la definición del lenguaje que acepta este autómata? Proporciona la gramática regular de los tokens que se reconocen.
- b) ¿Qué tokens son reconocidos al procesar la cadena 3e-z? Recuerda utilizar la técnica de la coincidencia más larga y si no es posible avanzar en un estado puedes hacer un retroceso o backtracking al estado de aceptación anterior para tratar de identificar el mayor número de tokens posible.

Solución.

a) A continuación se da la gramática requerida, esta es

– Desde el estado A.

$$\begin{aligned}
 A_e &\rightarrow iB_e \\
 &\rightarrow (a-h)D_e \\
 &\rightarrow (j-z)D_e \\
 &\rightarrow (A-Z)D_e \\
 &\rightarrow (.)D_e \\
 &\rightarrow (.)E_e \\
 &\rightarrow +F_e \mid -F_e \\
 &\rightarrow (0-9)G_e
 \end{aligned}$$

– Desde el estado B.

$$\begin{aligned}
 B_e &\rightarrow \epsilon \\
 &\rightarrow (a-e)D_e \\
 &\rightarrow (g-z)D_e \\
 &\rightarrow (A-Z)D_e \\
 &\rightarrow (.)D_e \\
 &\rightarrow (0-9)D_e \\
 &\rightarrow fC_e
 \end{aligned}$$

– Desde el estado C.

$$\begin{aligned}
 C_e &\rightarrow \epsilon \\
 &\rightarrow (a-z)D_e \\
 &\rightarrow (A-Z)D_e \\
 &\rightarrow (.)D_e \\
 &\rightarrow (0-9)D_e
 \end{aligned}$$

– Desde el estado D.

$$\begin{aligned}
 D_e &\rightarrow \epsilon \\
 &\rightarrow (a-z)D_e \\
 &\rightarrow (A-Z)D_e \\
 &\rightarrow (.)D_e \\
 &\rightarrow (0-9)D_e
 \end{aligned}$$

– Desde el estado E.

$$E \rightarrow (0-9)H_e$$

– Desde el estado F.

$$\begin{aligned}
 D_e &\rightarrow (.)E_e \\
 &\rightarrow (0-9)G_e
 \end{aligned}$$

– Desde el estado G.

$$\begin{aligned} G_e &\rightarrow \epsilon \\ &\rightarrow (0-9)G_e \\ &\rightarrow (.)H_e \\ &\rightarrow (eE)I_e \end{aligned}$$

– Desde el estado H.

$$\begin{aligned} H_e &\rightarrow \epsilon \\ &\rightarrow (0-9)H_e \\ &\rightarrow (eE)I_e \end{aligned}$$

– Desde el estado I.

$$\begin{aligned} I_e &\rightarrow +J|-J \\ &\rightarrow (0-9)k_e \end{aligned}$$

– Desde el estado J.

$$E \rightarrow (0-9)k_e$$

– Desde el estado k.

$$\begin{aligned} k_e &\rightarrow \epsilon \\ &\rightarrow (0-9)k_e \end{aligned}$$

Lo anterior es la gramática definida para los estados A¹, B, C, D, E, F, G, H, J, y K.

b) Para este inciso usemos el método visto en clase, así

1. Encontramos la cadena “3e-” del token pasando por los estados $[A \rightarrow G \rightarrow I \rightarrow J]$, al llegar a J realizamos backtracking para llegar a I, pues no encontramos una transición que nos permita encontrar z desde j. Caso análogo para I, G, y finalmente llegamos a el estado A.
2. Encontramos “e” por la transición $A \rightarrow D$. Sin embargo, no podemos encontrar el siguiente caracter del token y hacemos backtracking llegando nuevamente al estado inicial (A).
3. Encontramos “-” por la transición $A \rightarrow F$. Sin embargo, no podemos encontrar el siguiente caracter del token y hacemos backtracking llegando nuevamente al estado inicial (A).
4. Encontramos “z” por la transición $A \rightarrow D$. Sin embargo, no podemos encontrar el siguiente caracter del token y hacemos backtracking llegando nuevamente al estado inicial (A).

¹En la gramática se hace alusión a A_e para no confundir con el carácter A.