Manual técnico

Expresiones Regulares Para el desarrollo del analizador léxico se utilizaron las siguientes expresiones regulares para los distintos tokens que se ingresaran en el texto a analizar

{A-Z} +: Una palabra que contiene los caracteres de la A a la Z una o más veces

" {Cualquier Cosa excepto salto de línea} * ": Un string encerrado en comillas que contiene todo menos un salto de línea

(+|-)? {0-9]+: un numero entero positivo o negativo

(+|-)? {0-9]+. {0-9} +: expresión para número decimal.

{Símbolos validos}: Esta expresión contiene los símbolos:

"=" (Signo igual): 61

" [" (Corchete izquierdo): 91

"]" (Corchete derecho): 93

""" (Comillas dobles): 34

"," (Coma): 44

"{" (Llave izquierda o corchete): 123

"}" (Llave derecha o corchete): 125

";" (Punto y coma): 59

"(" (Paréntesis izquierdo): 40

")" (Paréntesis derecho): 41

"." (Punto): 46

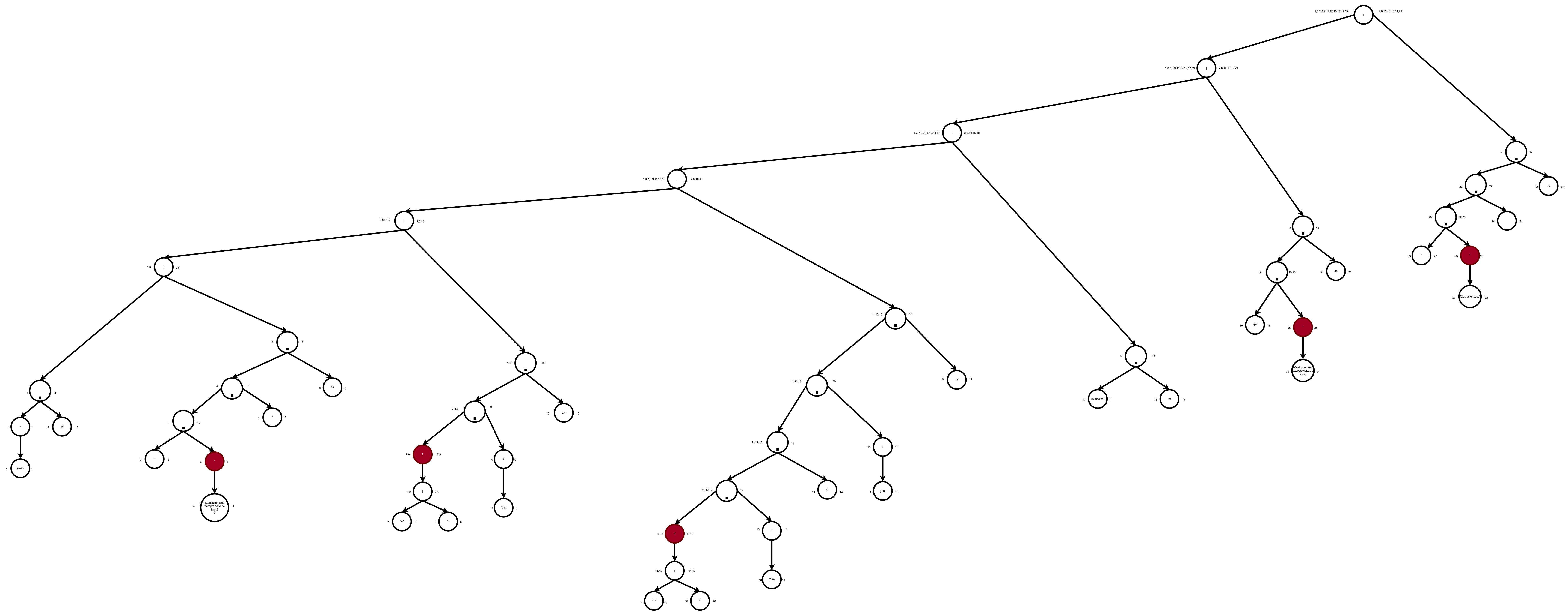
 # {Cualquier Cosa Menos Salto De línea}: expresión para los comentarios de una línea

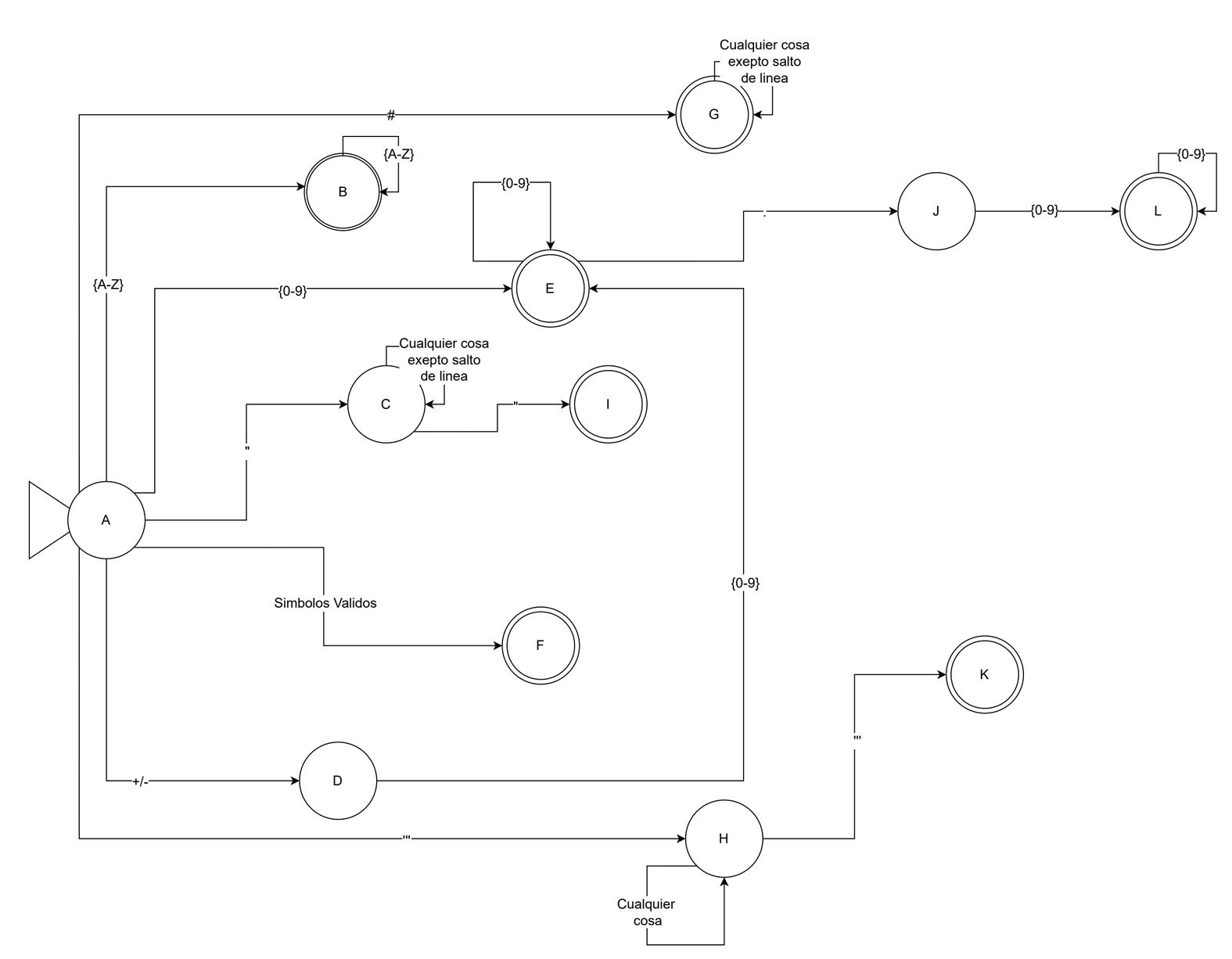
'''{Cualquier Cosa}''': expresión para los comentarios multilínea

Ahora se procede a construir la expresión regular completa para el Código:

{A-Z} + | " {Cualquier Cosa excepto salto de línea} * " | (+|-)? {0-9]+ | (+|-)? {09]+. {0-9} + | {símbolos validos} | # {Cualquier Cosa Menos Salto De línea} | '''{Cualquier Cosa}''' $

Y se realiza el árbol siguiente:




Cualquier cosa exepto salto de linea
#
G
B
{A-Z}
{0-9}
J
{0-9}
L
{0-9}
{A-Z}
{0-9}
E
Cualquier cosa exepto salto de linea
C
"
I
"
A
Simbolos Validos
F
{0-9}
K
+/-
D
H
Cualquier cosa

1 – {A-Z}	**1,2**
2 – 1#	
3 – “	4,5
4 – {Cualquier cosa, menos salto de línea}	4,5
5 – “	6
6 – 2#	
7 – “+”	9
8 – “-“	9
9 – {0-9}	9,10
10 – 3#	
11 –“+”	13
12 – “-“	13
13 – {0-9}	13,14
14 – “.”(punto)	15
15 – {0-9}	15,16
16 – 4#	
17 – {Símbolo valido}	18
18 – 5#	
19 – “#”	20,21
20 – {Cualquier cosa menos salto de línea}	20,21
21 – 6#	
22- ‘’’ (triple comilla)	23,24
23 – {Cualquier cosa}	23,24
24 – ‘’’	25
25 – 7#	

Tabla de follow del árbol

Y se realiza la tabla de transición

Estados	**{A-Z}**	**{Cualquier cosa excepto salto de línea}**	**“**	**+**	**-**	**{0-9}**	**.**	**{Símbolos validos}**	**#**	**{Cualquier cosa}**	**“”**
A = 1,3,7,8,9,11,12,13,17,19,22	B=1,2		C=4,5	D=9,13	D=9,13	E=9,10,13,14		F=18	G=20,21		H=23,24
B=1,2	B=1,2										
C=4,5		C=4,5	I=6								
D=9,13						E=9,10,13,14					
E=9,10,13,14						E=9,10,13,14	J=15				
F = 18											
G = 20,21		G=20,21									
H=23,24										H= 23,24	K=25
I=6											
J=15						L = 15,16					
K = 25											
L = 15,16						L= 15,16					

Tabla de transición generada a partir de la tabla de follow donde el estado 2,6,10,16,18,21 y 25 Son estados de aceptación

Y ya con esta tabla de transición

Se realiza el autómata a programar en el Código

```
Gramatica:

Terminales: Claves, igual, corcheteA, string, coma, corcheteC
           Registros, llaveA, llaveC, entero, decimal
           imprimir, parentesisA, parentesisC, puntocoma
           imprimirln, conteo, promedio, contarsi, max, min, exportarReporte
           datos, sumar

No Terminales:
<Inicio> <Claves> <ListaStrings> <Registros> <registro> <valor> <otroValor> <otroRegistro> <Funciones>
<funcion> <otrafuncion> <imprimir> <imprimirln> <conteo> <promedio> <contarsi>
<datos> <sumar> <max> <min> <exportarReporte>

Inicio:<Inicio>
Producciones:

    <Inicio> ::= <Claves> <Registros> <Funciones>

    <Claves> ::= Claves igual corcheteA string <ListaStrings> corcheteC
    <ListaStrings> ::= coma string <ListaStrings>
                    | lambda

    <Registros> ::= Registros igual corcheteA <registro> <otroregistro> corcheteC
    <registro> ::= llaveA <valor> <otroValor> llaveC
    <valor> ::= string
              | entero
              | decimal
    <otroValor> ::=  coma <valor> <otroValor>
                  | lambda
    <otroRegistro> ::= <registro> <otroRegistro>
                   | lambda

    <Funciones> ::= <funcion> <otrafuncion>
    <funcion>:: = <imprimir>
                |<imprimirln>
                |<conteo>
                |<promedio>
                |<contarsi>
                |<datos>
                |<sumar>
                |<max>
                |<min>
                |<exportarReporte>

    <imprimir> ::= imprimir parentesisA string parentesisC puntocoma
    <imprimirln> ::= imprimirln parentesisA string parentesisC puntocoma
    <conteo> ::= conteo parentesisA parentesisC puntocoma
    <promedio> ::= promedio parentesisA string parentesisC puntocoma
    <contarsi> ::= contarsi parentesisA string coma entero parentesisC puntocoma
    <datos> ::= datos parentesisA parentesisC puntocoma
    <sumar> ::= sumar parentesisA string parentesisC puntocoma
    <max> ::= max parentesisA string parentesisC puntocoma
    <min> ::= min parentesisA string parentesisC puntocoma
    <exportarReporte> ::= exporterReporte parentesisA string parentesisC puntocoma

    <otrafuncion> ::= <funcion><otrafuncion>
                   | lambda
```