Tecnológico de Monterrey

Grupo:

TC3002B.201

Materia:

Desarrollo de aplicaciones avanzadas de ciencias computacionales

Evidencia Final Compiladores: Desarrollo de herramienta de soporte al proceso de análisis de imágenes

Integrantes:

Salomon Dabbah Beracha - A01028445
Marco Antonio Torres - A01025334
Martín Palomares García - A01066569

Fecha de entrega:

03/05/24

Reglas de la gramática implementada

Rule 0 S' -> assignment
Rule 1 assignment -> VARIABLE SETTO expression
Rule 2 assignment -> VARIABLE SETTO flow
Rule 3 flow -> VARIABLE CONNECT flow_functions
Rule 4 flow_functions -> flow_function_call CONNECT flow_functions
Rule 5 flow_functions -> flow_function_call
Rule 6 flow_function_call -> VARIABLE LPAREN params RPAREN
Rule 7 assignment -> expression
Rule 8 expression -> expression PLUS term
Rule 9 expression -> expression MINUS term
Rule 10 expression -> term
Rule 11 expression -> string
Rule 12 string -> STRING
Rule 13 term -> term TIMES exponent
Rule 14 term -> term DIVIDE exponent
Rule 15 term -> exponent
Rule 16 exponent -> factor EXP factor
Rule 17 exponent -> factor
Rule 18 exponent -> LPAREN expression RPAREN
Rule 19 factor -> NUMBER
Rule 20 factor -> VARIABLE
Rule 21 factor -> function_call
Rule 22 function_call -> VARIABLE LPAREN RPAREN
Rule 23 function_call -> VARIABLE LPAREN params RPAREN
Rule 24 params -> params COMMA expression
Rule 25 params -> expression

Terminales

COMMA : 24
CONNECT : 3 4
DIVIDE : 14
EXP : 16
LPAREN : 6 18 22 23
MINUS : 9
NUMBER : 19
PLUS : 8
RPAREN : 6 18 22 23
SETTO : 1 2
STRING : 12
TIMES : 13
VARIABLE : 1 2 3 6 20 22 23
error :

Descripción de las funciones implementadas como herramientas y accesorios a la gramática

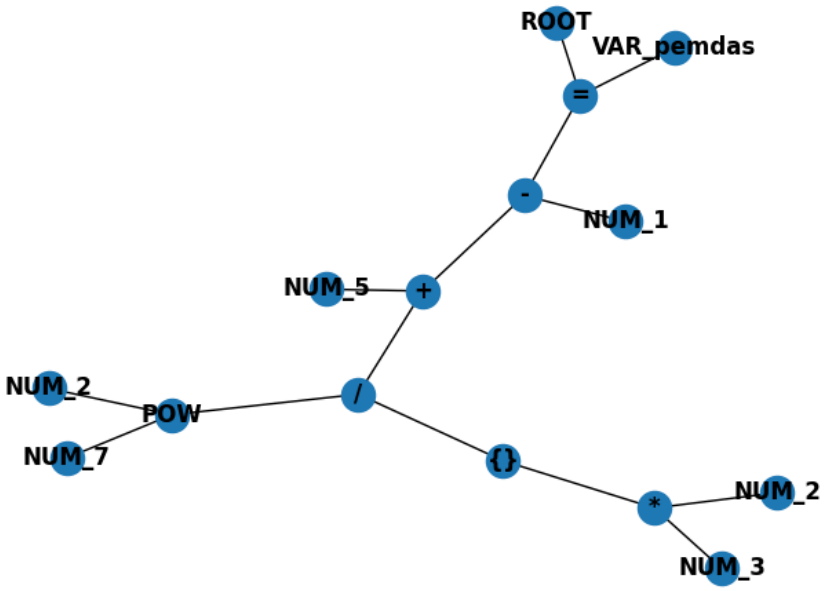
Función	Parámetro/s	Descripción
load_image	imagen	Carga una imagen al compilador y la lee.
save_image	nombre_del_archivo, imagen	Guarda la imagen en el dispositivo con el nombre seleccionado.
show_image	imagen	Despliega la imagen en el dispositivo.
search_cv2	nombre_de_la_funcion	Se usa para ver si una función existe en la librería de cv2, en caso de que exista, la función la regresa, de lo contrario arroja un error diciendo que no existe.
gen_matrix	filas, columnas, argumentos	Crea una matriz de x filas, por y columnas, y se le introducen los argumentos, es importante tener en cuenta que se tienen que introducir la cantidad de argumentos resultante de la multiplicación de x por y, por ejemplo, si la matriz sería de 3 x 3 tendría que tener 9 argumentos exactamente.
gen_vector	argumentos	Crea un vector que contiene los argumentos que se le proporcionen.
my_mean	argumentos	Obtiene la media de los argumentos proporcionados.
my_sum	argumentos	Obtiene la sumatoria de los argumentos proporcionados.
my_median	argumentos	Obtiene la mediana de los argumentos proporcionados.
my_std	argumentos	Obtiene la desviación estándar de los argumentos proporcionados.
my_max	argumentos	Obtiene el elemento más grande de los argumentos proporcionados.
my_min	argumentos	Obtiene el elemento más pequeño de los argumentos proporcionados.
my_sin	argumentos	Obtiene los senos de los argumentos proporcionados.
my_cos	argumentos	Obtiene los cosenos de los argumentos proporcionados.
my_tan	argumentos	Obtiene las tangentes de los argumentos proporcionados.
histogram	imagen	Convierte la imagen a escala de grises y calcula su histograma, que muestra la distribución de intensidades de píxeles de 0 (negro) a 255 (blanco).
canny_edge	imagen	Convierte la imagen a escala de grises y detecta los bordes utilizando el método de Canny. Luego, muestra la imagen original y la imagen de los bordes.
gray_scale	imagen	Carga la imagen y la convierte en blanco y negro.

Demostraciones

1. Precedencia de operadores

$$5 + (3 * 2)/7^2 - 1 = 202/49 = 4.1224489796$$

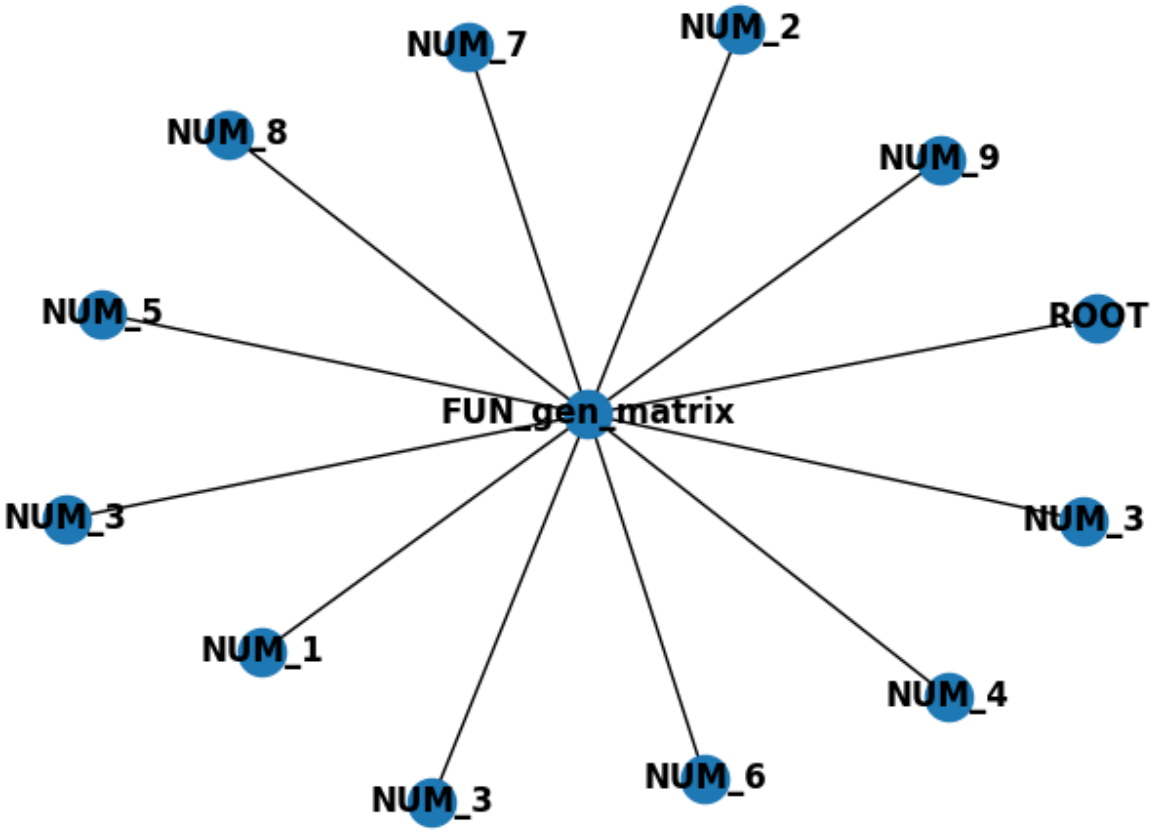
```
input>> pemdas = 5 + (3 * 2) / 7 ^ 2 - 1
From Node 14 []
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 2]
From Node 5 [6]
From Node 6 []
From Node 7 []
From Node 8 [7, 2]
From Node 9 [6, 49]
From Node 10 [5, 0.12244897959183673]
From Node 11 []
From Node 12 [5.122448979591836, 1]
From Node 13 ['pemdas', 4.122448979591836]
From Node 0 [4.122448979591836]
Result: 4.122448979591836
```



2. Llamadas a funciones

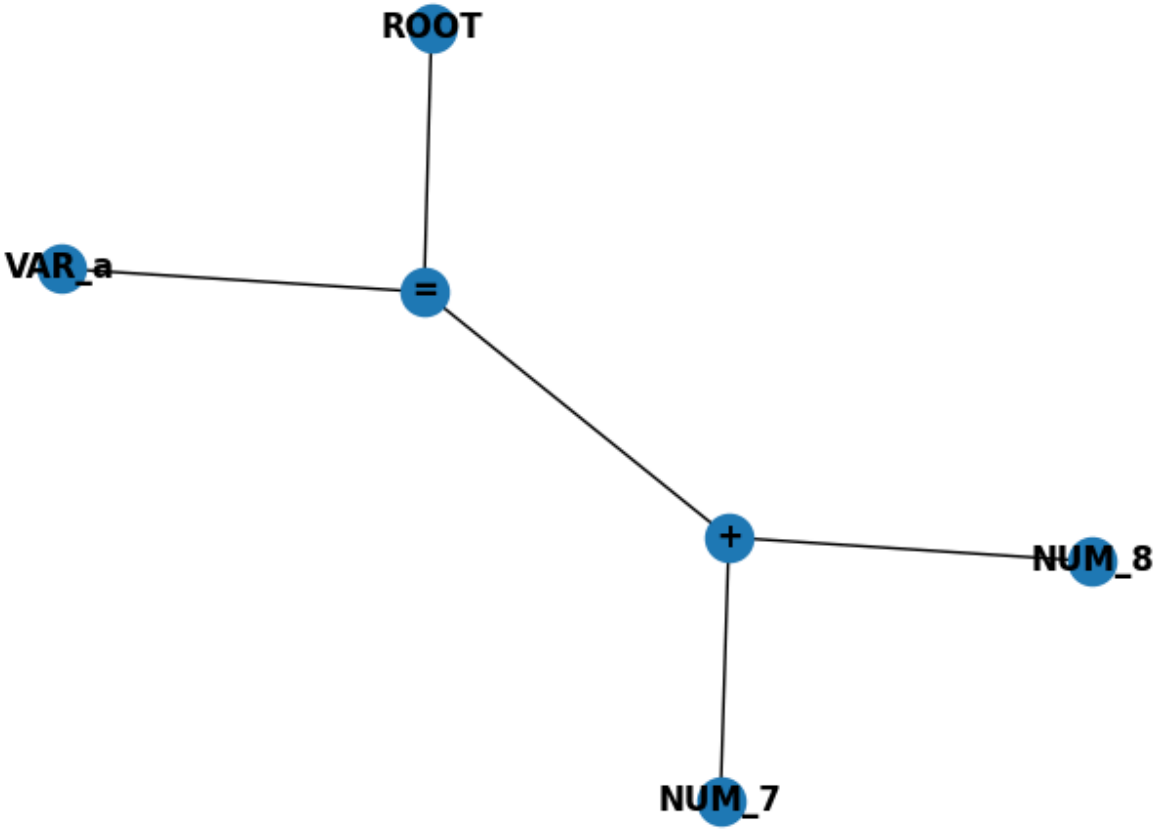
```
input>> gen_matrix(3,3,1,2,3,4,5,6,7,8,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_1', 'value': 1, 'counter': 3}, {'type': 'NUMBER', 'label': 'NUM_2', 'value': 2, 'counter': 4}, {'type'
: 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 5}, {'type': 'NUMBER', 'label': 'NUM_4', 'value': 4, 'counter': 6}, {'type': 'N
UMBER', 'label': 'NUM_5', 'value': 5, 'counter': 7}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 8}, {'type': 'NUMBE
R', 'label': 'NUM_7', 'value': 7, 'counter': 9}, {'type': 'NUMBER', 'label': 'NUM_8', 'value': 8, 'counter': 10}, {'type': 'NUMBER',
 'label': 'NUM_9', 'value': 9, 'counter': 11}]

From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 []
From Node 5 []
From Node 6 []
From Node 7 []
From Node 8 []
From Node 9 []
From Node 10 []
From Node 11 []
From Node 12 [3, 3, 1, 2, 3, 4, 5, 6, 7, 8, 9]
**** HERE Function call:  gen_matrix  ****
From Node 0 [array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])]
```



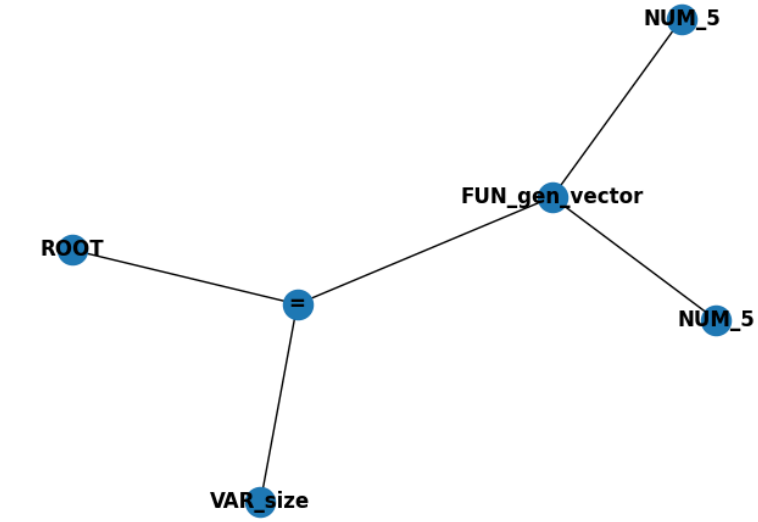
3. Asignación de variables

```
input>> a = 7 + 8
From Node 5 []
From Node 1 []
From Node 2 []
From Node 3 [7, 8]
From Node 4 ['a', 15]
From Node 0 [15]
Result: 15
```



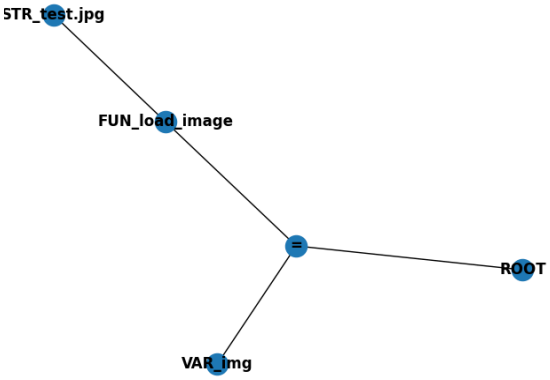
4. Implementación de flujos de imágenes y 5. Aplicación de filtros de Open CV

```
input>> size = gen_vector(5,5)
[{'type': 'NUMBER', 'label': 'NUM_5', 'value': 5, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_5', 'value': 5, 'counter': 2}]
From Node 5 []
From Node 1 []
From Node 2 []
From Node 3 [5, 5]
**** HERE Function call:  gen_vector  ****
From Node 4 ['size', array([5, 5])]
From Node 0 [array([5, 5])]
```




```
input>> img = load_image("test.jpg")
[{'type': 'STRING', 'label': 'STR_test.jpg', 'value': 'test.jpg', 'counter': 1}]
From Node 4 []
From Node 1 []
From Node 2 ['test.jpg']
**** HERE Function call:  load_image  ****
From Node 3 ['img', array([[[252, 243, 233],
        [252, 243, 233],
        [252, 243, 233],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       [[253, 244, 234],
        [253, 244, 234],
        [253, 244, 234],
        ...,
```

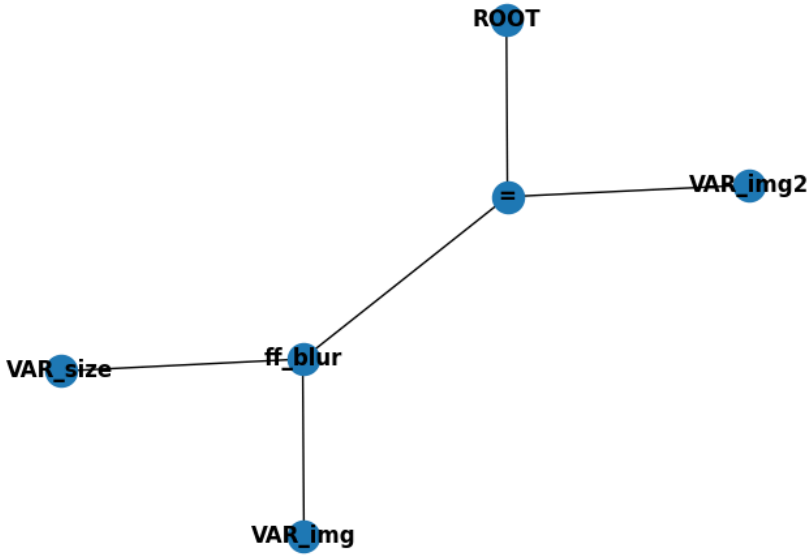


```
input>> img2 = img -> blur(size)
From Node 5 []
From Node 3 []
From Node 1 []
From Node 2 [array([[[252, 243, 233],
        [252, 243, 233],
        [252, 243, 233],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       [[253, 244, 234],
        [253, 244, 234],
        [253, 244, 234],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       [[254, 244, 234],
        [254, 244, 234],
        [254, 244, 234],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       ...,
```




```
input>> show_image(img2)
[{'type': 'VARIABLE', 'label': 'VAR_img2', 'value': 'img2', 'counter': 1}]
From Node 1 []
From Node 2 [array([[[253, 244, 234],
        [253, 244, 234],
        [253, 244, 234],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       [[253, 244, 234],
        [253, 244, 234],
        [253, 244, 234],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       [[254, 244, 234],
        [254, 244, 234],
        [254, 244, 234],
        ...,
        [  1,   2,   0],
        [  1,   2,   0],
        [  1,   2,   0]],

       ...,
```