Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes Formales y Autómatas
Sección: 02
Ing. Campos Gonzáles Vivian Damaris

Proyecto 2 - Fase 1

Método del Árbol

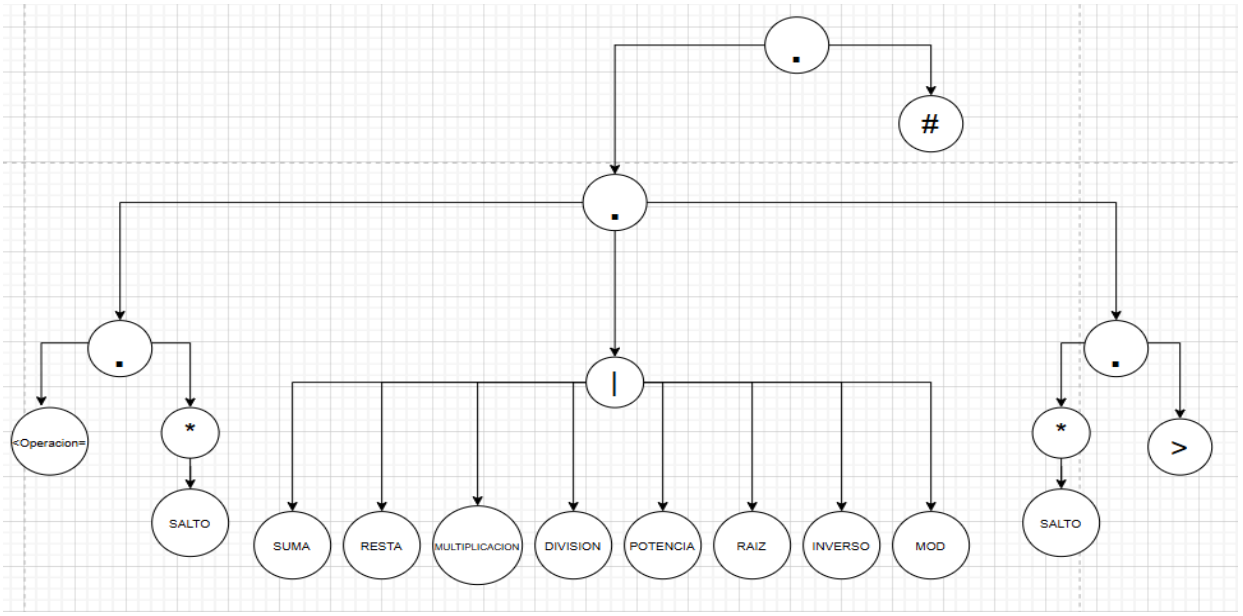
Zane Christian Bran Lockridge - 1373724
Esteban Josué Maldonado Velasco - 1053424

Guatemala 22 de Octubre del 2025

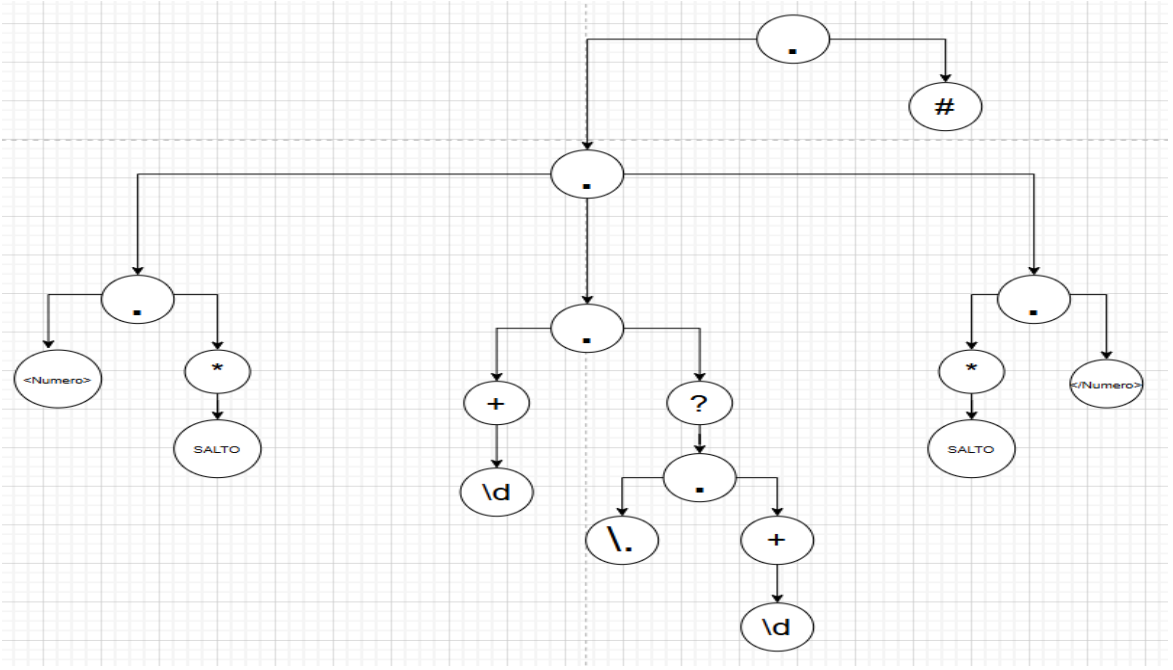
Árbol de expresión

Una vez definidas las Expresiones Regulares y sus respectivos tokens, se procedió a construir los árboles sintácticos individuales de cada uno. En cada caso, se concatena el token con el símbolo #, el cual es indispensable dentro del método del árbol, ya que representa el fin de cadena o final de la expresión regular. Este símbolo permite que, al aplicar las funciones del método, el autómata resultante pueda reconocer la finalización de cada lexema y realizar una transición válida hacia un estado de aceptación.

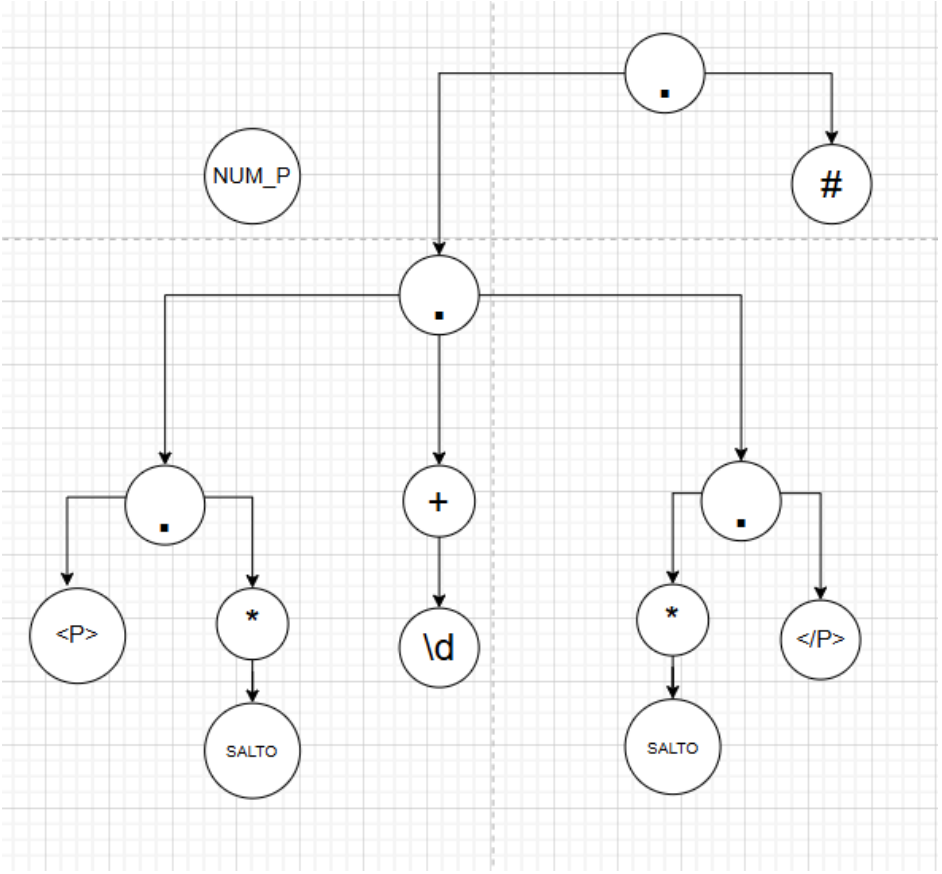
Árbol del Token OPE_A



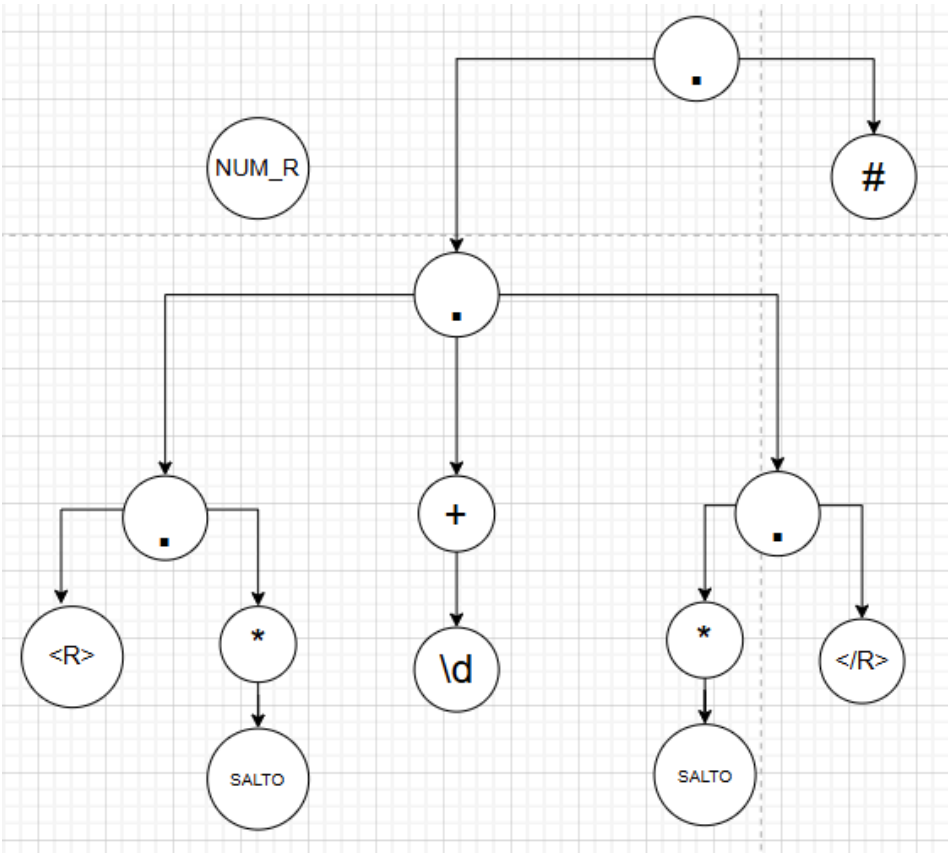
Árbol del Token NUM



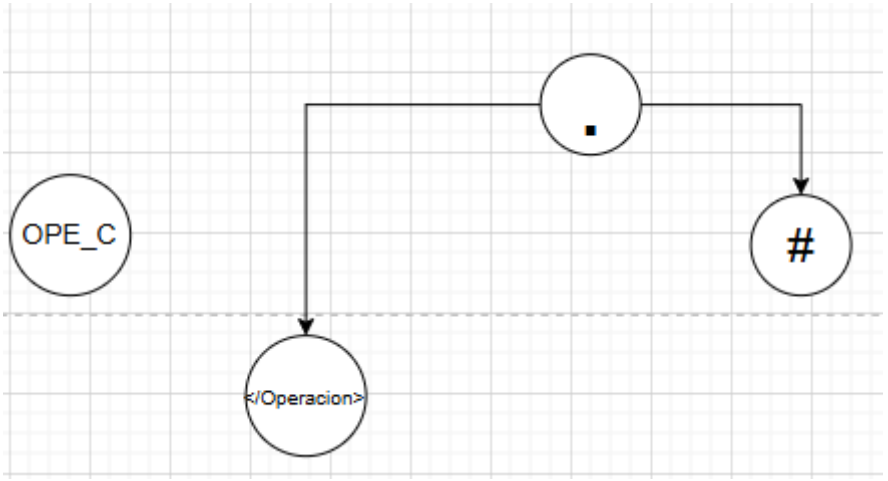
Árbol del Token NUM_P



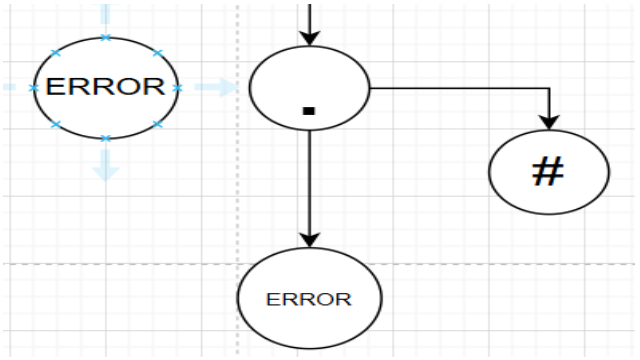
Árbol del Token NUM_R



Árbol del Token OPE_C

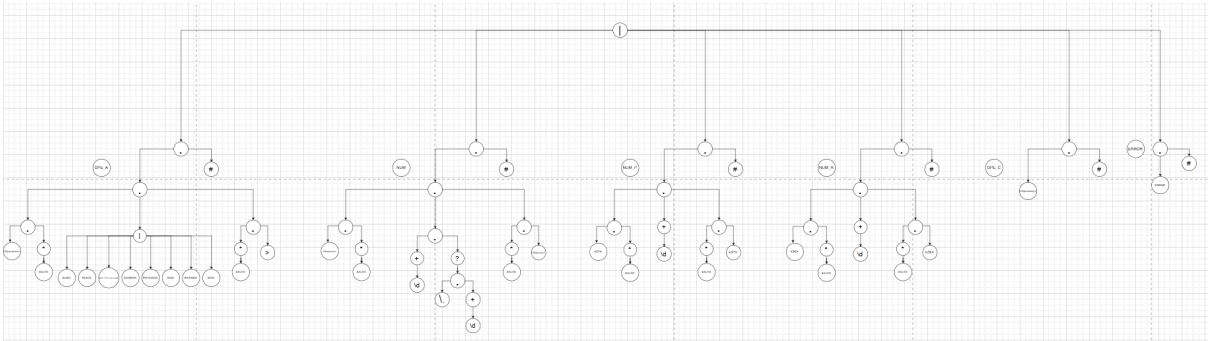


Árbol del Token ERROR



Se unificaron los árboles individuales en un árbol general, en el cual cada token se combina mediante el operador de unión (|).
A partir de este árbol general se calcularon las tablas de funciones, la tabla de transiciones y finalmente se construyó el autómata determinista finito correspondiente al analizador léxico.

Árbol General



La representación gráfica completa del árbol general se encuentra disponible en el archivo “Árboles.png”, incluido en el repositorio oficial del proyecto en GitHub, donde puede visualizarse con mayor detalle la estructura y relaciones entre los tokens.

Cálculo de funciones

La estrategia metodológica adoptada para el análisis léxico del proyecto se sustenta en dos decisiones de diseño que buscan la eficiencia, la compacidad y la claridad formal del autómata resultante: el tratamiento de los tokens como unidades terminales indivisibles y la implementación de un único símbolo de fin de cadena #.

1. Tratar lo tokens como hojas

En el contexto del proyecto, se ha considerado cada token (OPE_A, NUM, NUM_P, NUM_R, OPE_C y ERROR) como una unidad léxica terminal. Esta aproximación se justifica por las siguientes razones técnicas y de diseño:

- El objetivo primordial de la fase léxica es construir un autómata capaz de reconocer y clasificar dichos tokens. El análisis de la estructura interna carácter por carácter de cada token se considera una preocupación que excede el alcance de esta etapa, evitando así la necesidad de modelar la composición interna de cada lexema.

- Representar cada token como una hoja en el árbol de expresiones regulares simplifica notablemente la construcción del árbol general. Esta abstracción previene la explosión combinatoria que ocurriría si se intentara modelar carácter por carácter expresiones léxicas extensas como por ejemplo: <Operacion= SUMA> o <Operacion= MULTIPLICACION> , manteniendo la estructura del autómata más compacta y manejable.

2. Un único símbolo #

Se ha optado por emplear un único símbolo de fin de cadena # concatenado al árbol general de expresiones, en lugar de un símbolo individual por cada token alternativo. Esta elección es completamente consistente con el método de construcción de autómatas a partir de árboles y ofrece ventajas operativas:

- El símbolo # funciona como el marcador de la terminación del lexema reconocido por el autómata general. Esto permite una definición precisa de los estados de aceptación del Autómata Finito Determinista, que corresponden a aquellos conjuntos de posiciones que incluyen la posición de #.

- El uso de un único # asegura que el DFA resultante sea capaz de reconocer la finalización de la lectura de cualquier token alternativo definido en el lenguaje. Además, facilita y estandariza el cálculo de la tabla follow y la construcción de la tabla de transiciones.

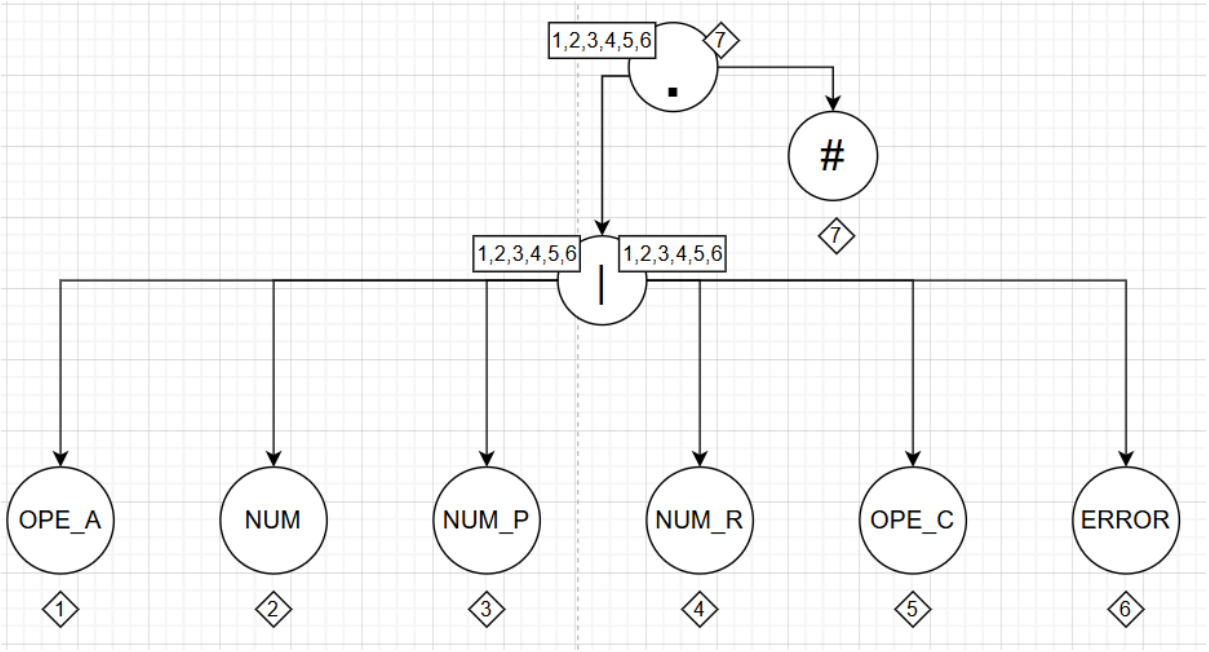


Tabla de follow

1	OPE_A	7
2	NUM	7
3	NUM_P	7
4	NUM_R	7
5	OPE_C	7
6	ERROR	7
7	#	—

Tabla de transiciones

Conjunto	Símbolo	Nuevo Conjunto	Estado de Aceptación
A = {1,2,3,4,5,6}	OPE_A	B = {7}	NO
A = {1,2,3,4,5,6}	NUM	B = {7}	NO
A = {1,2,3,4,5,6}	NUM_P	B = {7}	NO
A = {1,2,3,4,5,6}	NUM_R	B = {7}	NO
A = {1,2,3,4,5,6}	OPE_C	B = {7}	NO
A = {1,2,3,4,5,6}	ERROR	B = {7}	NO
B = {7}	#	—	SI

En el desarrollo del analizador léxico, la tabla de transiciones obtenida mediante el método del árbol refleja que todas las transiciones desde el estado inicial conducen a un único estado de aceptación. Este resultado se debe al propósito del autómata construido: reconocer tokens individuales, no la estructura completa del lenguaje.

En esta expresión, cada token válido del lenguaje se representa como una hoja independiente dentro del árbol sintáctico, unidas entre sí por el operador de unión |. Por tanto, cada símbolo puede aparecer de manera independiente dentro del texto de entrada.

Durante la construcción del autómata, el único operador de concatenación existente se encuentra al final de la expresión #, lo que implica que todas las hojas del árbol están seguidas directamente por el símbolo de fin de cadena. Aplicando la regla de follow para el nodo de concatenación, se obtiene que todas las posiciones correspondientes a los tokens tienen como follow únicamente el conjunto {7}, donde la posición 7 corresponde al símbolo #

En consecuencia, al generar la tabla de transiciones:

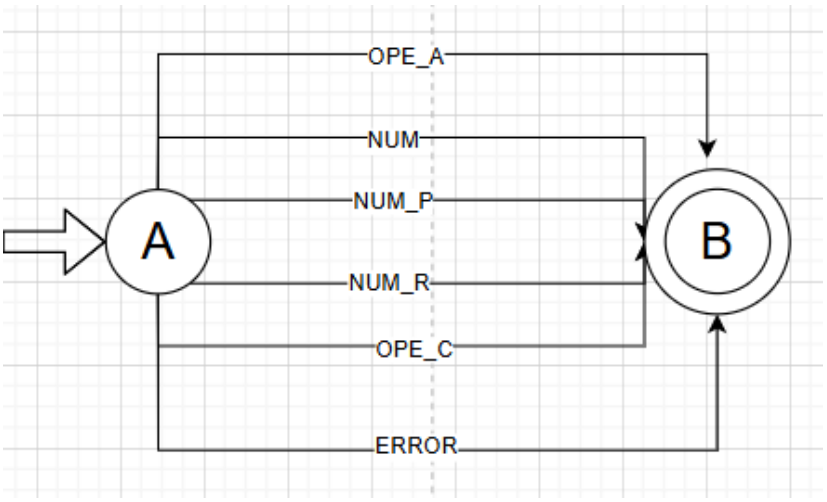
- El estado inicial (A) contiene todas las posiciones asociadas a los tokens (1,2,3,4,5,6)

- Desde este estado, al leer cualquiera de los símbolos válidos el autómata se desplaza hacia un único estado de aceptación (B), que contiene la posición {7}.

- No existen más transiciones posteriores, ya que el símbolo # marca el fin del reconocimiento.

De esta manera, el DFA generado no representa la secuencia de tokens en el código fuente, sino que reconoce cada token por separado, identificando cuándo un lexema válido ha sido completamente leído.

DFA



De acuerdo con las especificaciones establecidas en el documento oficial del proyecto, la Fase I se limita al desarrollo del análisis léxico. En esta etapa, el objetivo es aplicar el método del árbol a las expresiones regulares diseñadas para los diferentes tokens, obteniendo a partir de ellas la tabla de follow, la tabla de transiciones y el autómata finito determinista (DFA) resultante.

El propósito de este autómata es reconocer tokens válidos e identificar errores léxicos en el archivo de entrada, tal como se indica en el apartado de entregables: “Código fuente del escáner funcional, generado con base a la tabla de transiciones, que permite leer, validar e identificar los lexemas, tokens y errores del archivo de entrada.”

Por lo tanto, en esta fase no se considera el análisis sintáctico ni la validación estructural de las operaciones aritméticas, ya que dichos aspectos corresponden a la Fase II, donde se implementa la interpretación completa del lenguaje y la jerarquía de las operaciones.