Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes Formales y Autómatas
Ing. Moisés Alonso

PROYECTO LENGUAJES FORMALES Y AUTÓMATAS
FASE 2

Integrantes:
Megan Naómi Morales Betancourt 1221120
Emilio Antonio Barillas Contreras 1150620
Roberto Alfredo Moya Noack - 1273020

Fecha de entrega: 29 de marzo del 2023

INTRODUCCIÓN

En el presente trabajo, se realiza una documentación técnica detallada del generador de scanner, que tiene como objetivo automatizar el proceso de análisis léxico en la construcción de compiladores. El entregable de este proyecto consiste en la generación de las tablas de First, Last y Follow, así como la tabla de transiciones del autómata finito determinista correspondiente.

El análisis léxico y el análisis sintáctico son etapas cruciales en la construcción de un compilador y de un autómata, ya que se encarga de identificar los diferentes elementos léxicos que conforman el código fuente de un programa. Estos elementos léxicos son los tokens, que representan las palabras clave, operadores, identificadores, constantes y otros elementos del lenguaje de programación.

El generador de scanner implementa un enfoque basado en autómatas finitos deterministas (AFD), que permite una eficiente identificación de los tokens. Para lograr esto, el generador de scanner utiliza las tablas de First, Last y Follow.

La tabla de First contiene la información sobre el primer conjunto de terminales que pueden aparecer en una cadena derivada de un símbolo no terminal. La tabla de Last, por su parte, indica el último conjunto de terminales que pueden aparecer en una cadena derivada de un símbolo no terminal. Finalmente, la tabla de Follow describe los terminales que pueden aparecer inmediatamente después de un símbolo no terminal en una cadena derivada.

La tabla de transiciones del autómata finito determinista representa el conjunto de estados y las transiciones entre ellos, y es generada a partir de las tablas de First, Last y Follow. Esta tabla es fundamental para la implementación eficiente del scanner, ya que permite la identificación rápida de los tokens mediante la lectura de caracteres del código fuente.

ANÁLISIS

Para poder iniciar con el proceso de la realización del árbol, ya que nuestro árbol recibe una lista de tokens, necesitamos general la lista con las concatenaciones y los '|' debidos. Para empezar, se empiezan a recorrer las líneas e ir validando cada carácter, agregando las concatenaciones entre tokens y cada '|' al final de cada línea, haciendo las respectivas validaciones (no concatenar si es un símbolo no terminar, etc.).

Al final del recorrido, agregar los siguientes caracteres '). #'. Cada token se va almacenando de una lista de tokens llamada 'regular expresion'. Por último, se envía la lista al árbol para que este se genere.

Para iniciar, se necesita un listado tokens, una vez este generado hace las siguientes validaciones:

Primero se itera sobre todos los tokens, luego va leyendo el siguiente token, si el token es un símbolo terminal, se creará un nuevo nodo para él y se apilará en la pila S. Si el token es un paréntesis de apertura, se apila en la pila T. Si el token es un paréntesis de cierre, llama la función "paréntesis_cierre" que realizará lo siguiente:

- Primero va a des apilar los símbolos de la pila T hasta encontrar el paréntesis de apertura. Si el siguiente token en la pila T tiene la misma prioridad que el token actual, lo des apilará y creará un nuevo nodo.

Saliendo de esa función se sigue con el recorrido de los tokens, si el token es un operador llama a 'operadores unarios', crea un nuevo nodo para el operador y lo apila en la pila S. Luego des apila el último nodo en esta misma pila y lo asigna primero al hijo izquierdo del nuevo nodo creado. Si no es un operador unario como la concatenación y el OR, si la pila T no está vacía y el sig. token tiene más prioridad, se des apila y se crea un nuevo nodo y se usan como hijos los últimos dos nodos.

Este proceso se sigue mientras la pila T no esté vacía. Luego se llama la función de TablaLastFollow para empezar con la creación de las tablas.

Una vez terminado el árbol se envía a otra clase la cual se encargará de obtener las tablas de Follow y Transiciones.

Con el árbol se recorre de forma "PostOrder", esto con el objetivo de que visite las hojas primero y luego el padre. Se hace un primer sondeo en Post Order en donde se enumeran las hojas y cada nodo no hoja, se determina si es Nullable o no. Los nodos no hoja, serán los operadores por lo que según; que operador és, se determinará si es Nullable o no por medio de un bool.

Una vez hecho esto se vuelve a recorrer una vez más el árbol en PostOrder para trabajar con cada nodo no hoja o nodo que contenga una operación. Para cada evaluación se revisará qué operación és y en base a eso se determinará su First y Last.

Seguido de eso, se vuelve a realizar un recorrido del árbol, pero esta vez para construir la tabla de Follow en cada nodo que contenga una operación válida. Con la tabla de Follow hecha, se procede

a realizar la tabla de transiciones la cual comienza con el First del nodo raíz. Este primer estado se comparará con cada uno de los elementos terminales o el dato de las hojas para ver si tiene relación y si es así, se tomará su correspondiente a la tabla de Follow. Esto sucederá para todos los elementos terminales y si no existe el estado, se agregará a una lista para que vuelva a ser iterada.

Por último, se imprime cada una de las tablas. La tabla de Follow con la cantidad de símbolos terminales y sus estados y la tabla de transiciones con sus estados y transiciones a cada símbolo terminal

DIAGRAMAS

Diagrama Tokens

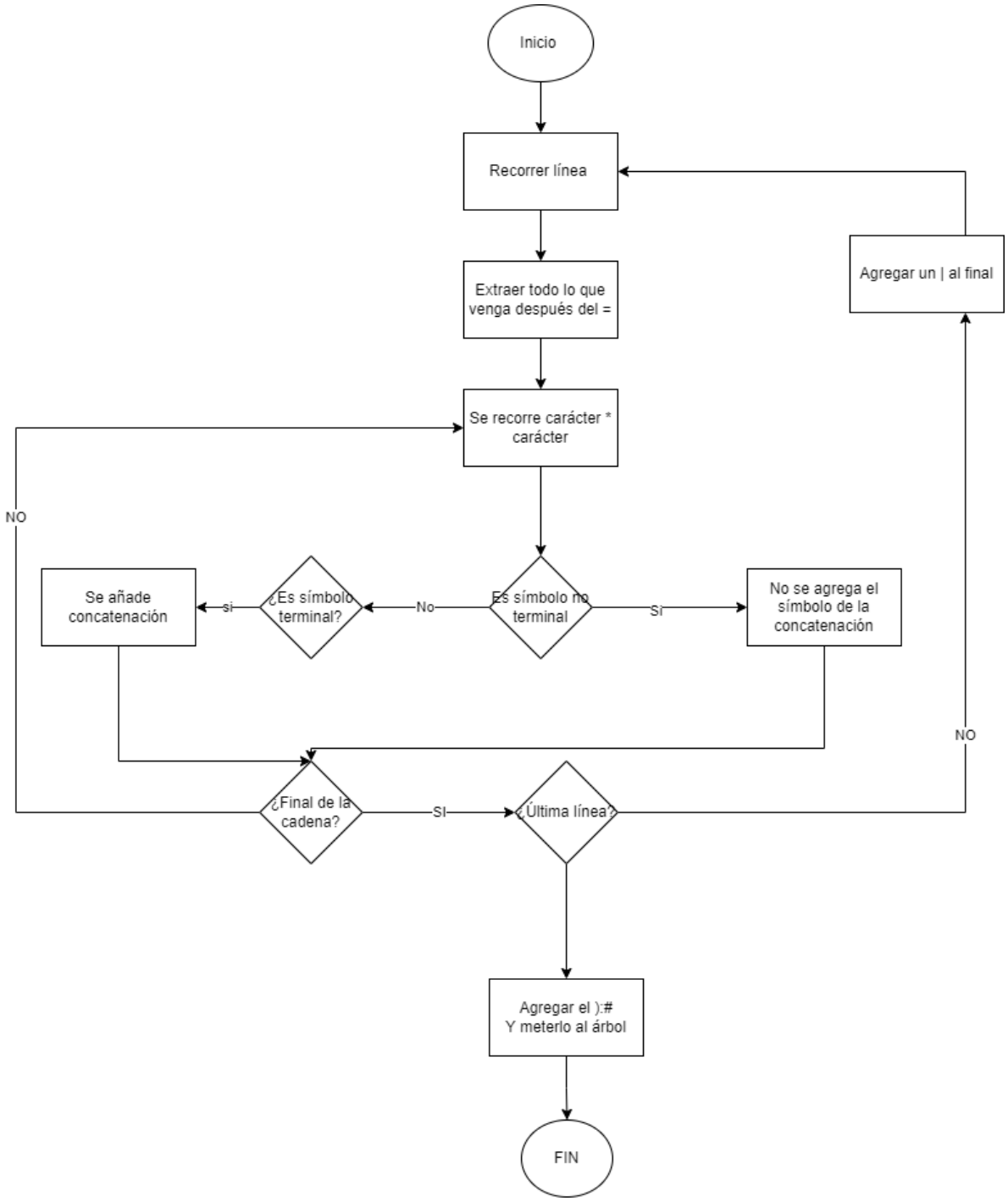

Diagrama 1: Diagrama de separación de tokens

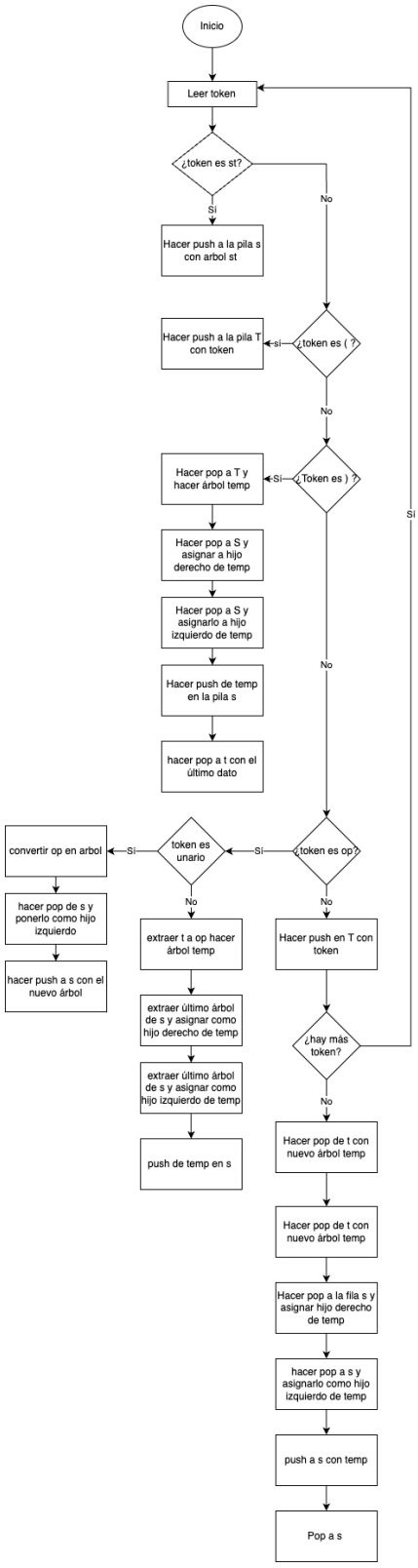

Diagrama 2: Diagrama del árbol de expresión

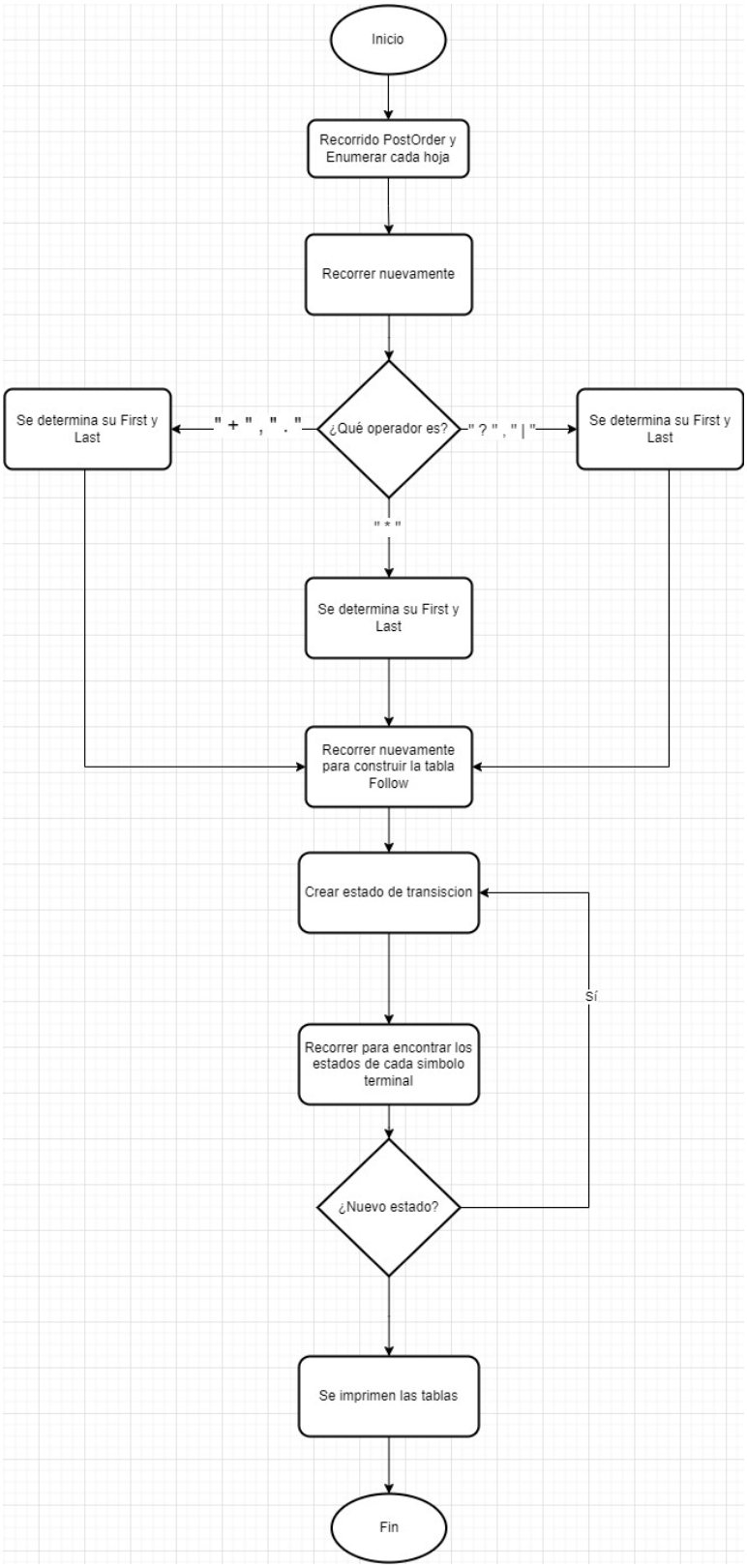

Diagrama No.3 – Diagrama First, Last, Nullable, Tabla Follow y Transiciones.

CONCLUSIONES

El generador de scanner permite una implementación eficiente del análisis léxico y sintáctico, lo que reduce significativamente el tiempo y los recursos necesarios para esta etapa en el proceso de construcción de un compilador.

Al utilizar un enfoque basado en autómatas finitos deterministas y las tablas de First, Last y Follow, se logra una identificación precisa y rápida de los tokens, lo que mejora el rendimiento y la fiabilidad del autómata resultante.

El generador de scanner es una herramienta que puede ser aplicada no solo en la construcción de compiladores, sino también en otras áreas relacionadas con el procesamiento de lenguaje natural y la identificación de patrones en textos. Su eficiencia y precisión lo convierten en una opción viable y efectiva para diversas aplicaciones.

ANEXOS

Manual de usuario:

1. Al darle Run, hace la validación de los SET – TOKEN – ACTIONS

```
SET valido linea 3
SET valido linea 4
TOKEN valido linea 6
TOKEN valido linea 7
TOKEN valido linea 8
TOKEN valido linea 9
TOKEN valido linea 10
TOKEN valido linea 11
TOKEN valido linea 12
TOKEN valido linea 13
TOKEN valido linea 14
TOKEN valido linea 15
TOKEN valido linea 16
TOKEN valido linea 17
TOKEN valido linea 18
TOKEN valido linea 19
TOKEN valido linea 20
TOKEN valido linea 21
TOKEN valido linea 22
TOKEN valido linea 23
TOKEN valido linea 24
TOKEN valido linea 25
TOKEN valido linea 26
TOKEN valido linea 27
TOKEN valido linea 28
TOKEN valido linea 29
TOKEN valido linea 30
TOKEN valido linea 31
TOKEN valido linea 32
TOKEN valido linea 33
TOKEN valido linea 34
TOKEN valido linea 35
TOKEN valido linea 36
```

2. Saca la tabla de first, last y follow con respecto al archivo de entrada

```
Tabla follow

0 |
1 | 2,56
2 | 2,56
3 | 4
4 | 5
5 | 56
6 | 7
7 | 8
8 | 56
9 | 56
10 | 11
11 | 56
12 | 56
13 | 56
14 | 15
15 | 56
16 | 17
17 | 56
18 | 56
19 | 56
```

3. Saca la tabla de las transiciones

```
Tabla de Transiciones          |DIGITO|              |'"'|          |CHARSET|            |'''|          |'='|          |'<'|          |'>'|     |
'+'|          |'-'|          |'0'|          |'R'|          |'*'|          |'A'|          |'N'|          |'D'|          |'M'|          |'I'|     |
'V'|          |'T'|          |'('|          |')'|          |';'|          |'.'|          |'{'|          |'}'|          |'['|          |']'|     |
':'|          |','|          |LETRA|
S0: {1,3,6,9,10,12,13,14,16,18,19,20,22,23,26,29,32,35,37,39,40,41,42,43,44,45,46,47,49,50,51,53} |     | {2,56} |      | {4} |        | {} |  | {7} |    |
 {56} |        | {11,17,56} |  | {15,56} |     | {56} |       | {56} |       | {21} |       | {} |  | {38,56} |    | {24} |       | {33} |       | {30} |    | {27} |       | {} |  | {} |
S1: {2,56} |    | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S2: {4} |       | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S3: {} |        | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S4: {7} |       | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S5: {56} |      | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S6: {11,17,56} |        | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S7: {15,56} |   | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S8: {21} |      | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S9: {38,56} |   | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S10: {24} |     | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S11: {33} |     | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S12: {30} |     | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S13: {27} |     | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
S14: {36,56} |  | {54,55,56} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |  | {} |
```