BFN to Railway (No editable)

Cantu Sanchez Abraham Isai*
Pena Cuellar Aldo de Jesus*
Torres Colorado Juan Daniel*
Trevino Gandarilla Jesus David*
*Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

***Resumen*—Este documento mostrará el proceso del proyecto "BNF To Railway"[1], el cual fue desarrollado como un proyecto de equipo en el contexto de la Asignatura de Lenguajes y Autómatas, utilizando el lenguaje de programación C++[2] tomando en cuenta la biblioteca de Qt 5.[3] El objetivo principal del proyecto es crear un programa con una interfaz interactiva con el usuario capaz de generar de forma dinámica un diagrama a partir de la entrada de texto que el usuario proporcione en una notación predefinida.**

I. Introducción

En el presente documento del proyecto llamado "BFN to Railway (No editable)" se detalla el trabajo realizado y asignado como proyecto de equipo en la asignatura de Lenguajes y Autómatas en el lenguaje de programación C++ [4] utilizando la libreria Qt5 [3]. El propósito principal de este proyecto tiene como meta ralizar un programa con una interfaz interactiva que cuente con la capacidad de realizar dinámicamente un diagrama apartir desde una entrada de texto y de la notación predefinida.

La *BNF* (Notación de Backus-Naur) [5] es un método formal usado para describir la sintaxis de un lenguaje de programación que es entendido como *Backus Naus Formas*. Diferentes lenguajes tienen diferentes descripciones y reglas, pero la estructura general de un BNF es:

- '`nombre ::= expansión`': Significa que puede "expandirse hacia" y "puede ser reemplazado por".
- '`<>`': Todos los nombres son rodeados por corchetes angulares.
- '`function / +`': Un símbolo terminal puede ser una literal.
- '`|`': Indica elección.

Los *Railway/Railroad Diagram* (Diagrama de Ferrocarril) [6] son usados para representar la estructura de un lenguaje como un conjunto de diagramas, en dónde cada diagrama representa una parte del lenguaje y juntos representan completamente el lenguaje. En cada diagrama empieza desde el extremo izquierdo, se divide en diferentes caminos que pueden cruzarse y finalmente convergen en un punto final en el extremo izquierdo.

II. Desarrollo Experimental

Para iniciar, se hizo una investigación exhaustiva de información a través de diversos materiales empleados tanto por parte de elementos previamente impartidos durante la asignatura, como lo son las diapositivas Compilers de Pierre Geurts [7], así como de investigación propia por parte de los integrantes de este proyecto. Esto se realizó con el propósito de tener una mejor compresión y entendimiento sobre concepto para llevar a cabo este proyecto, tales como el análisis léxico[8] y de sintaxis[9].

Durante el desarrollo del proyecto, se hizo necesario desarrollar una sintaxis BNF en la cuál fuera capaz de operar con los requisitos necesarios para llegar a esta finalidad. Dicha sintáxis es:

```
- definition
      =
      :=
      ::=
- concatenation
      ,
      <whitespace>
- termination
      ;
- alternation
      |
- option
      [ ... ]
      ?
- repetition
       ...  => 0..N
      expression* => 0..N
      expression+ => 1..N
      <digits>* expression => <digits>...<digits>
      <digits>* [expression] => <0>...<digits>
      <digits>* expression? => <0>...<digits>
- grouping
      ( ... )
- literal
      '' ... '' or ' ... '
- special characters
      (? ... ?)
```

Esta sintaxis incluye elementos como definiciones, concatenaciones, terminaciones, alternaciones, opciones, repeticiones, agrupaciones, literales y caracteres especiales. Dicha sintaxis proporcionada, servirá para la analizar la cadena ingresada en el campo establecido para ello y en dado caso que dicha cadena es válida, entonces se demostrará posteriormente su diagrama. Esto se manejará directamente en la ejecución del programa, como se ve en la figura 1.

III. Resultados

Figura 1: Ejecución

El texto ingresado en la aplicación puede ser utilizado para generar un diagrama para la sintaxis de una consulta SQL [10] utilizando la sentencia SELECT.

En la figura 2 hay un campo de texto con el esquema de la consulta SQL, que incluye varias cláusulas como SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, UNION, LIMIT y OFFSET. Cada una de estas cláusulas está representada en el diagrama de flujo por debajo del campo de texto, donde se organizan en bloques rectangulares conectados.

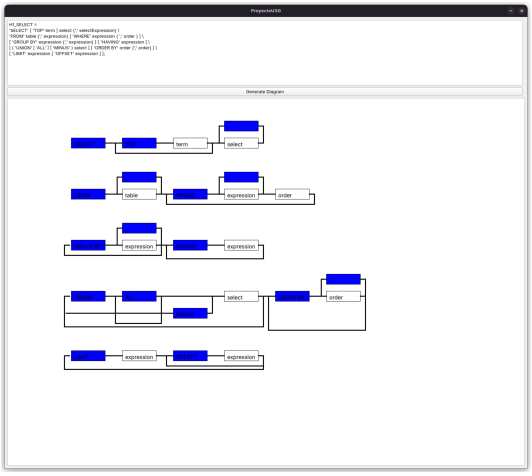

Figura 2: Ingreso de cadena y diagrama generado. Ejemplo 1

La parte superior de la figura 3 tiene un área de texto con la sintaxis de la sentencia SQL, incluyendo palabras clave como DISTINCT, ALL, UNION, LIMIT, OFFSET, SAMPLE_SIZE, y FOR UPDATE. Debajo, el diagrama de flujo estructura visualmente cómo se pueden combinar estas cláusulas en una consulta real. Por ejemplo, la cláusula SELECT puede ser seguida por TOP, DISTINCT, o ALL, indicando diferentes modos de seleccionar los datos.

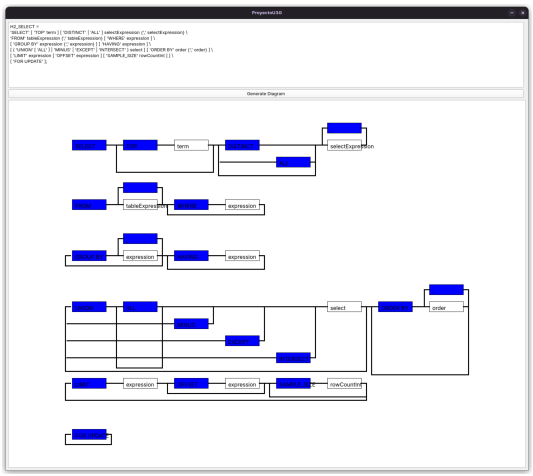

Figura 3: Ingreso de cadena y diagrama generado. Ejemplo 2

En la figura 4, se tiene una sintaxis en la cual describe detalladamente cómo debe estructurarse una instrucción SELECT o DELETE, proporcionando flexibilidad en términos de qué cláusulas y expresiones pueden ser incluidas o no, y en qué orden. Esta sintaxis sería útil para implementar un analizador sintáctico que pueda entender y procesar consultas SQL en un sistema de gestión de bases de datos o en un compilador SQL.

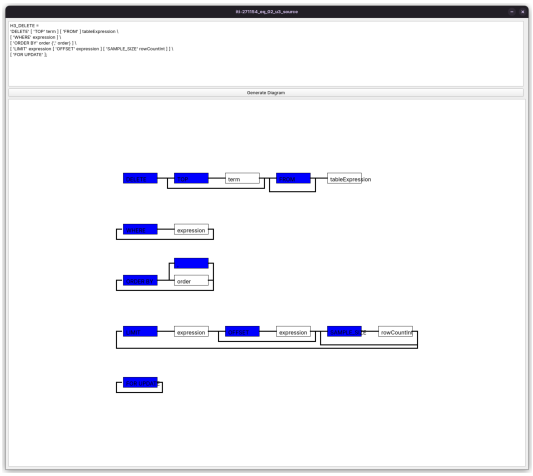

Figura 4: Ingreso de cadena y diagrama generado. Ejemplo 3

IV. Conclusión

En conclusión, el proyecto ”BFN to Railway”ha sido un esfuerzo colaborativo para desarrollar una aplicación de escritorio que cumple con el objetivo de convertir gramáticas definidas en notación BNF en diagramas de ferrocarril de manera dinámica. Utilizando el lenguaje de programación C++ y la biblioteca Qt5, se ha creado una interfaz interactiva que permite a los usuarios ingresar sus gramáticas y obtener visualizaciones gráficas claras y comprensibles de las mismas.

A lo largo del desarrollo del proyecto, se ha logrado implementar con éxito la generación de diagramas, la gestión de agrupaciones dentro de corchetes y la visualización adecuada de las producciones gramaticales. Aunque el proceso de desarrollo ha presentado desafíos y obstáculos, el equipo ha trabajado de manera colaborativa para superarlos y alcanzar los objetivos establecidos.

El programa resultante ofrece una herramienta útil para estudiantes y profesionales que trabajan con gramáticas formales, facilitando la comprensión y visualización de estructuras gramaticales complejas. Además, el proyecto ha brindado una oportunidad invaluable para aplicar los conocimientos teóricos adquiridos en la asignatura de Lenguajes y Autómatas en un contexto práctico y relevante.

Referencias

[1] Michael L. Scott. *Programming Language Pragmatics*. 2000.

[2] y Barbara E. Moo Stanley B. Lippman Josée Lajoie. *Ç++ Primer"*. Addison-Wesley Professional, 2012.

[3] The Qt Company. *QT 5*. https://doc.qt.io/qt-5/qt5-intro.html. Consultado el 17-04-2024.

[4] Michael T. Goodrich & Roberto Tamassia & David Mount. *Data Structures & Algorithms in C++*. Second Edition. 2011, págs. 4-10.

[5] J. S. Rohl. “A note on Backus Naur form”. En: *The Computer Journal* 10.4 (1968), págs. 334-341.

[6] Andreas Leon Aagaard Moth. *A Software Tool for Learning Syntax*. Technical University of Denmark, Informatics y Mathematical Modelling, 2011.

[7] Pierre Geurts. *Compilers*. http://www.montefiore.ulg.ac.be/~geurts/compil.html. Consultado el 17-04-2024.

[8] Unidentified. *Análisis léxico*. https://docs.python.org/es/3/reference/lexical_analysis.html. Consultado el 17-04-2024.

[9] Arrlex. *Guía Completa sobre la Sintaxis en Programación: Aprende y Domina*. https://arrlex.com/que-es-sintaxis-en-programacion/. Consultado el 17-04-2024.

[10] Tihomir Babic. *Sintaxis SQL*. https://learnsql.es/blog/sintaxis-sql/. Consultado el 17-04-2024.