Universidad Rafael Landívar

Ingeniería en Informática y Sistemas

Lenguajes Formales y Autómatas

Ing: Moisés Alonso

***Proyecto #2***

Cristian González – 1183221

Luis Girón – 1188021

Guatemala 29 de marzo del 2023

Introducción

En esta fase del proyecto se nos solicitó que una vez que se lograra leer el documento creáramos un árbol de expresiones usando como base los datos en el área de tokens, para esto cada uno de los tokens señalados serán separados por alternaciones, además la expresión resultante será colocada entre paréntesis y concatenada con el símbolo terminal ‘#’ para determinar el final de esta.

Una vez se obtenga esta expresión se colocara en una función que creara un árbol de expresiones, esto nos servirá para obtener los datos de los símbolos para las tablas de First, Last y Follow. Los datos de estas tablas son la que nos servirían para crear las tablas de transiciones del autómata infinito, además de crear los estados.

Para realizar esto se creó un proyecto en Visual Studios con el lenguaje C# con el formato de Windows Forms.

# Bitácora

Lo primero que se realizo fue trasladar lo ya realizado en el primer proyecto a un nuevo formato con una mejor interfaz gráfica, usando en este caso Windows Forms, después de realizar esto se procedió a crear la expresión regular que sería introducida al árbol, para esto se recorrería el archivo y al encontrar una línea con el formato “TOKEN #= CARACTER”, tomaría el carácter lo agregaría a otro cadena y le colocaría al final una alternación | esto se repetiría con todas las líneas de tokens, cabe resaltar que al existir la posibilidad de que aparezcan los símbolo .,\* o | como terminales además de los paréntesis, estos serían remplazados con números que posteriormente serían devueltos a su valor original. Además se encerraría entre paréntesis y se concatenaría con el símbolo terminal #

Una vez que tenemos la expresión regular, se crearía dos clases para poder crear el árbol de expresiones, una clase Nodo con los direccionamientos a sus hijos, el dato que se guarda, la nullabilida, el first y el last. Usando esta clase nodo se crearía la clase de árbol, para esto se modificó un árbol de expresiones aritméticas, se le incluyeron instrucciones para sus paréntesis, se adaptaron los símbolos que se usan para determinar que va en cada árbol, y se creó un método que guarda un solo símbolo para los símbolos unitarios.

Para poder asignar los first, last y follows solo fue necesario recorrer un par de veces el árbol, una para asignar los valores a las hojas, otra para dar los valores a los separadores y la última vez para la tabla follow. Esto sería de utilidad para la tabla de transiciones.

Expresiones Regulares

Trasladada del Proyecto 01

1. SETS(\\n)\*(([A-Z])+=((('([A-Z]|[a-z]|[0-9])'\\.\\.'([A-Z]|[a-z]|[0-9])'|'\_')(\\+)?)+(\\n)\*| CHR\\(([0-9])\*\\)\\.\\.CHR\\(([0-9])\*\\)(\\n)?))+(\\n)\*)?(\\n)\*
   1. Esta parte de la expresión evalúa la presencia del campo SETS, de existir evaluaría si cumple con las condiciones que indicadas como la presencia de un nombre, y que los paréntesis se cierren.
2. TOKENS(\\n)\*(TOKEN[0-9]\*\\=([A-Z]+|\\{RESERVADAS\\(\\)([A-Z]\*\\(\\))\*\\}|\\((([A-

Z])+|\\||\\\*|\\+)+\\)|(\\\*|\\+|\\|)?|\\'(\\\"|[A-Z]|\\'|\\=|\\<|\\>|\\+|\\|\\\*|\\(|\\)|\\;|\\.|\\{|\\}|\\[|\\]|\\:|\\,)\\')+(\\n)\*)+

1. Esta expresión evalúa el campo de Tokens, asegura que existe, además de tomar en cuente que alas comillas se abran y cierran, acepta un símbolo a la ves y se pueden repetir varias veces.
2. ACTIONS(\\n)\*RESERVADAS\\[(\\)(\\n)+\\{\\n+([0-9]\*\\=\\'[A-Z]+\\'\\n+)+\\}\\n+([AZ]\*\\(\\)\\n+\\{\\n+([0-9]\*\\=\\'[A-Z]\*\\'\\n+)+\\}\\n+)\*[A-Z]](file://///)(/n)+/%7b/n+(%5b0-9%5d*/=/'%5bA-Z%5d+/'/n+)+/%7d/n+(%5bA-Z%5d*/(/)/n+/%7b/n+(%5b0-9%5d*/=/'%5bA-Z%5d*/'/n+)+/%7d/n+)*%5bA-Z)
   1. Esta parte de la expresión evalúa la presencia del campo ACTIONS asegurándose que exista evaluaría si cumple con las condiciones que indicadas como la presencia de Reservadas, la numeracion, y que los paréntesis se cierren.
3. \*ERROR\\=[0-9]+\\n\*?"
   1. Esta parte de la expresión evalúa la presencia del campo ERROR asegurando que exista al menos un número presente.

Expresión que verifica que la línea contiene un Token

^(\\s)\*TOKEN(\\s)\*[0-9]+(\\s)\*\\=