Conceptos Básicos y Algoritmos Fundamentales

Lexer Aritmético: expresiones regulares y autómatas en Scheme Racket

Bo Hyeon Cha, A01023804
Juan Carlos Garfias Tovar, A01652138

Luis Humberto González Guerra

14 de marzo del 2020

Propósito del programa

Los analizadores léxicos o analizadores lexicográficos son la primera fase de un compilador. Reciben como entrada una secuencia de caracteres y dan como salida componentes léxicos. Estos tokens posteriormente son usados para la traducción y posteriormente para el analizador sintáctico.

Manual del Usuario

Para poder hacer uso del programa creado es necesario cumplir con los siguientes requisitos:

- Windows 10 x64 bits versión 20H2 (Compilación SO 19042.867) o superior
- Racket 8.0 (Febrero 2021)
- DrRacket

El programa recibe como entrada un archivo de texto que contiene expresiones aritméticas y comentarios. Regresando la tokenización de los elementos en el archivo en el orden en que fueron encontrados e indicando el tipo de elemento.

La versión actual del Lexer aritmético cuenta con las siguientes expresiones:

- Enteros
- Flotantes (Reales)
- Operadores:
    - Asignación
    - Suma
    - Resta
    - Multiplicación
    - División
    - Potencia
- Identificadores:
    - Variables
- Símbolos especiales:
    - (
    -)
- Comentarios:
    - // seguido de caracteres hasta que se acabe el renglón

Este tokenizador funciona a partir de espacios. Es decir que toma como elemento cada string que esté separada por espacio. Es decir que para que el programa tome correctamente los elementos se necesitan separar todos los elementos por espacio. Sin embargo, el programa también soporta saltos de línea por lo que los saltos de línea y espacios están soportados por el tokenizer.

El programa cuenta con varias funciones, las cuales fungen como elementos de clasificación booleana a partir de expresiones regulares tanto de tipo #rx como de tipo #px. Esto permite hacer la clasificación de manera directa a partir de patrones, parámetros y condiciones dadas. De esta manera son clasificados todos los elementos de los textos a analizar. El programa evalúa el string en cada uno de estos elementos y en caso de no ser válido ninguno regresa un mensaje de "Error de formato". De esta manera el programa es capaz de detectar anomalías semánticas y de clasificar con precisión cada uno de los elementos.

Para poder usar el código es necesario abrir el archivo en DrRacket. Posteriormente se da click del lado superior derecho en el botón de run. Los archivos a analizar deben de estar en formato .txt en el mismo directorio donde se encuentre el programa. Posteriormente saldrá el siguiente mensaje:

```
Welcome to DrRacket, version 8.0 [cs].
Language: racket, with debugging; memory limit: 128 MB.
Bohyeon Cha & Juan Carlos Garfias
Ingresar (lexerAritmetico Nombre del archivo)
>
```

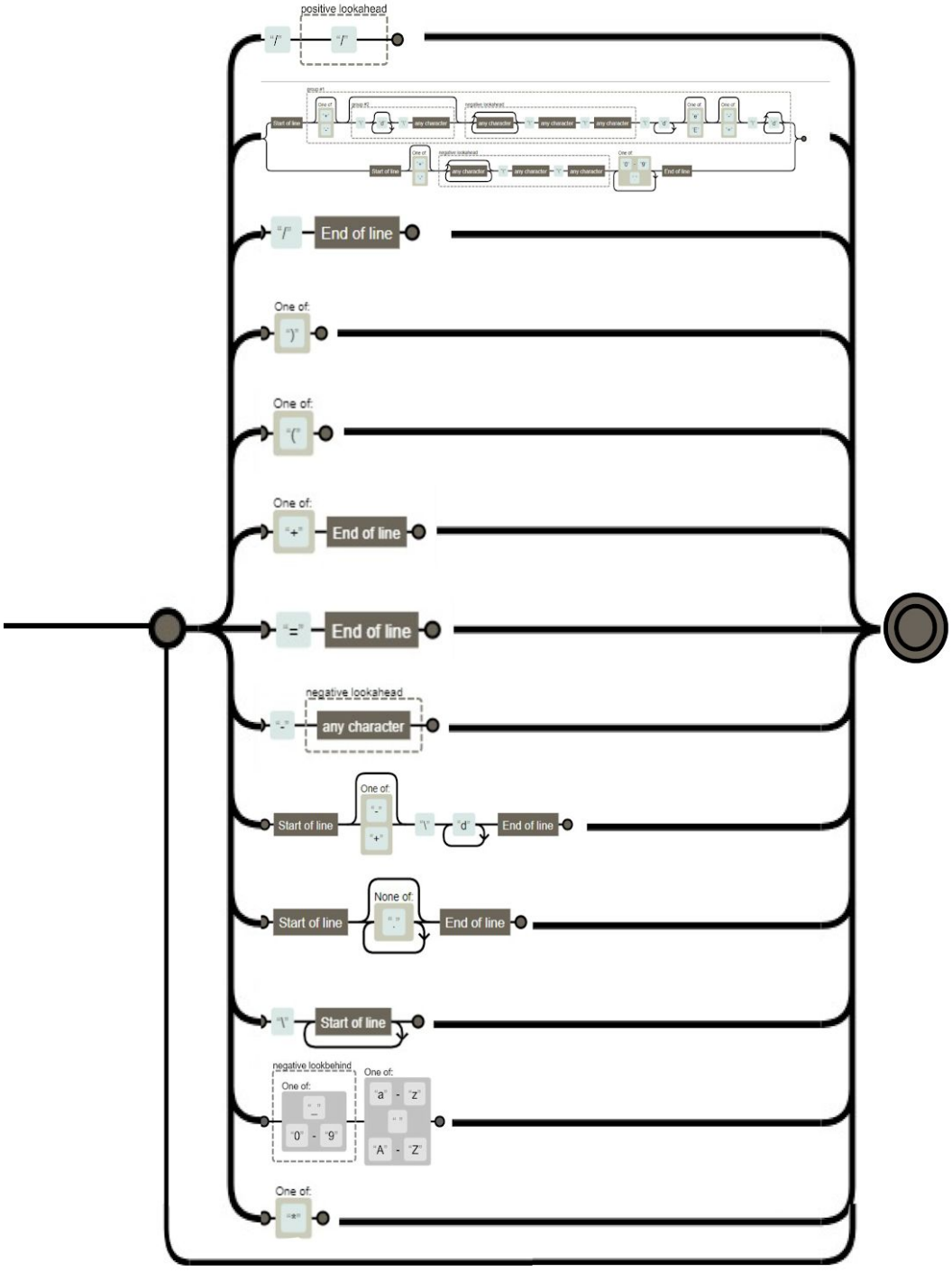
Una vez completado este output significa que el programa ha cargado todos los métodos necesarios para realizar la tokenización. Posteriormente el usuario puede ingresar (lexerAritmetico "archivo.txt") reemplazando el nombre por el archivo deseado. El programa regresará el siguiente output correspondiente al análisis correspondiente.

```
Bohyeon Cha & Juan Carlos Garfias
Ingresar (lexerAritmetico Nombre del archivo)
> (lexerAritmetico "05.txt"
                   )

Token   Tipo
ex_a    Variable
=       Asignacion
45.3    Real
+       Suma
10      Entero
ex_b    Variable
=       Asignacion
aber    Variable
-       Resta
(       Parentesis que abre
10.4    Real
+       Suma
        Error de formato
6.1E-8  Real
)       Parentesis que cierra
*       Multiplicacion
23      Entero
10.0    Real
+       Suma
10.     Real
/       Division
.23     Real
variable_3      Variable
=       Asignacion
2.3E3   Real
/       Division
6.345e-5        Real
*       Multiplicacion
-0.001E-3       Real
-       Resta
.467E9  Real
230..   Error de formato
>
```

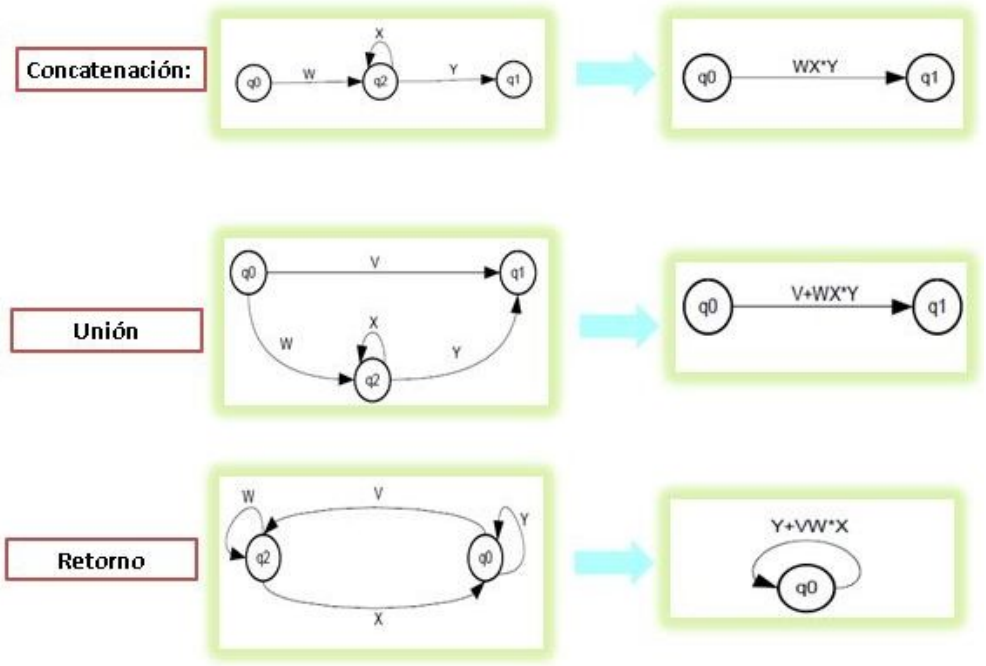
Funcionamiento lógico del programa

El programa funciona a partir de procesos de un sistema determinista. Es decir que el sistema está basado en un autómata finito determinista, en el cual existen diferentes estados, de los cuales en caso de cumplirse avanzan o regresan a un estado específico. A continuación se muestra el diagrama de estados creado a partir de los requerimientos lexicográficos.



Este autómata posteriormente fue simplificado y transformado en expresiones regulares. Las cuales son una secuencia de caracteres que conforman un patrón de búsqueda. Este tipo de sistemas pueden ser considerados como los símbolos que componen a sus transiciones. Cuando se elimina alguno de sus estados se reemplazan los cambios que

pasaban a través de él como transiciones directas pasan a ser con ingresos de expresiones regulares en lugar de símbolos. A continuación se muestran ejemplos de estas conversiones:



De esta forma se logró simplificar todos los procesos en expresiones regulares, las cuales hacen el acceso a los patrones más directos y evitan tantos niveles de recursión, los cuales a la larga, especialmente en textos de gran cantidad una mayor complejidad algorítmica y tiempo de ejecución. De esta manera el sistema pasa a evaluar una menor cantidad de estados, verificando únicamente si son comentarios, divisiones, paréntesis, asignaciones,sumas,restas, multiplicaciones,potencias,reales, enteros,variables u otros elementos. De esta manera la recursión únicamente se realiza por línea y luego por elemento. De la manera tradicional se habría realizado por línea, luego por elemento y finalmente por carácter. Añadiendo así otra capa de recursión que tendría que verificar elementos anteriores y futuros, los cuales ya son directamente evaluados con el uso de expresiones regulares.

Referencias

4.8 Regular Expressions. (2021). Retrieved 15 March 2021, from https://docs.racket-lang.org/reference/regexp.html

(2021). Retrieved 15 March 2021, from https://ccc.inaoep.mx/~emorales/Cursos/Automatas/ExpRegulares.pdf

(2021). Retrieved 15 March 2021, from https://www.institucional.frc.utn.edu.ar/sistemas/ghd/T-M-AFD.htm

Autómatas finitos y expresiones regulares - TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES. (2021). Retrieved 15 March 2021, from https://sites.google.com/site/wikiudocsctalf/home/vision-de-las-expresiones-regulares-y-lenguajes-regulares/automatas-finitos-y-expresiones-regulares