INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Implementación de métodos computacionales

Evidencia 1

Vladimir Mendoza

Julián Enrique Espinoza Valenzuela A01254679

20, marzo del 2024

Lexical_Analyzer_Documentation

Manual de usuario:

- Paso 1:

Para poder utilizar el analizador léxico llevado a cabo, es necesario que en primer lugar se importen las librerías que vienen adjuntas en el código proporcionado. Las librerías son las siguientes:

import Data.Char (isSpace, isDigit, isLetter, isAlpha, isAlphaNum)

import Data.List.Split (splitOn)

import Data.List.Utils (replace)

import Data.List (stripPrefix, isPrefixOf, isInfixOf,elemIndex)

import Data.Maybe (catMaybes)

- Paso 2:

Ya teniendo instaladas las librerías anteriormente mencionadas es necesario tener en cuenta que debemos de tener presente dos cosas principalmente: la primera es que debemos contar con un archivo .txt que podamos dar como alimento al programa. Cabe destacar este archivo debe de estar en el mismo directorio que se encuentre su código guardado y debe de estar incluido específicamente en la siguiente parte del código para que todo funcione como debe de ser:

```
121  main :: IO ()
122  main = do
123      content <- readFile "algebraic_expresions.txt"
124
125      let linesOfText = lines content
126      mapM_ processLine linesOfText
```

En la parte donde dice “algebraic_expresions.txt” usted puede poner el archivo .txt que contenga sus expresiones algebraicas o en su defecto puede utilizar el archivo .txt incluido en la entrega. En segundo lugar, se debe resaltar que si se agrega un archivo .txt distinto al que viene con la entrega se debe asegurar de que venga una expresión algebraica por línea para que todo funcione como se espera.

- Paso 3:

Teniendo lo anterior en cuenta, lo único que resta es especificar los comandos necesarios para compilar el programa y correrlo. En este caso y debido al nombre del archivo serían los siguientes:

-Para compilar:

ghc -o lexycal_analyzer lexycal_analyzer.hs

-Para correr el código:

./lexical_analyzer

Casos de prueba:

1. Se da como entrada una expresión algebraica con una variable bien definida y un entero:
   Input:

```
3    val_4=4
```

   Output:

```
val_4 Variable

= Assignment Operator

4 Integer
```

2. Se da como entrada una variable mal definida y un número real:

   Input:

```
4    9_bro=18.5
```

   Output:

```
9_bro Lexycal error(Variable name has a wrong composition)
= Assignment Operator

18.5 Real
```

3. Se da como entrada una variable bien definida con números reales no válidos, notaciones exponenciales validas y no validas:

   Input:

```
7    a = 5e9 *(-8.6.4 - b)/       6.1E-8.4
```

   Output:

```
5e9 Real
* Multiplication Operator
( Opening Parenthesis
- Subtraction Operator
8.6.4 Lexycal error()-------------------------FIX
- Subtraction Operator
b Variable
) Closing Parenthesis
```

```
/ Division Operator

6.1E Real
- Subtraction Operator
8.4 Real

6.1E-8.4 Lexycal error()-------------------------FIX2
```

4. Se da como entrado una variable valida con un numero real y un comentario:

   Input:

```
a=   8.   //   bro1
```

   Output:

```
a Variable

= Assignment Operator

8. Real

//bro1 Comment
```

5. Se da como entrada un comentario:

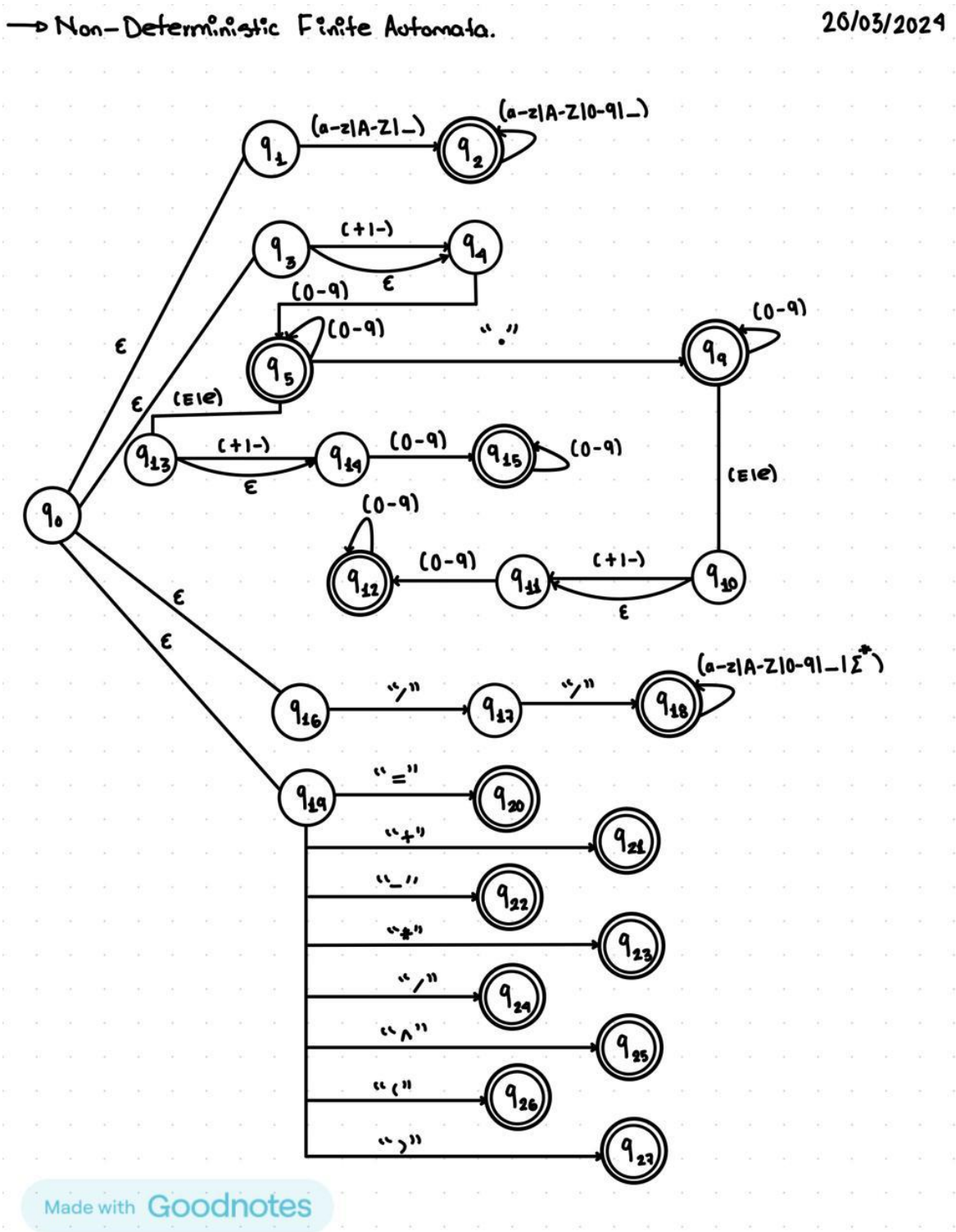
   Input:

```
1   //This is a comment
```

   Output:

```
//This is a comment Comment
```

NDFA:



Descripción de conjuntos de estados:

Variables: {q0, q1, q2}

Enteros y reales: {q0, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12}

Comentarios: {q13, q14, q15}

Operador “+”: {q0, q16, q18}

Operador “-”: {q0, q16, q19}

Operador “=”: {q0, q16, q17}

Operador “*”: {q0, q16, q20}

Operador “^”: {q0, q16, q22}

Signo “(”: {q0, q16, q23}

Signo “)”: {q0, q16, q24}

DFA:

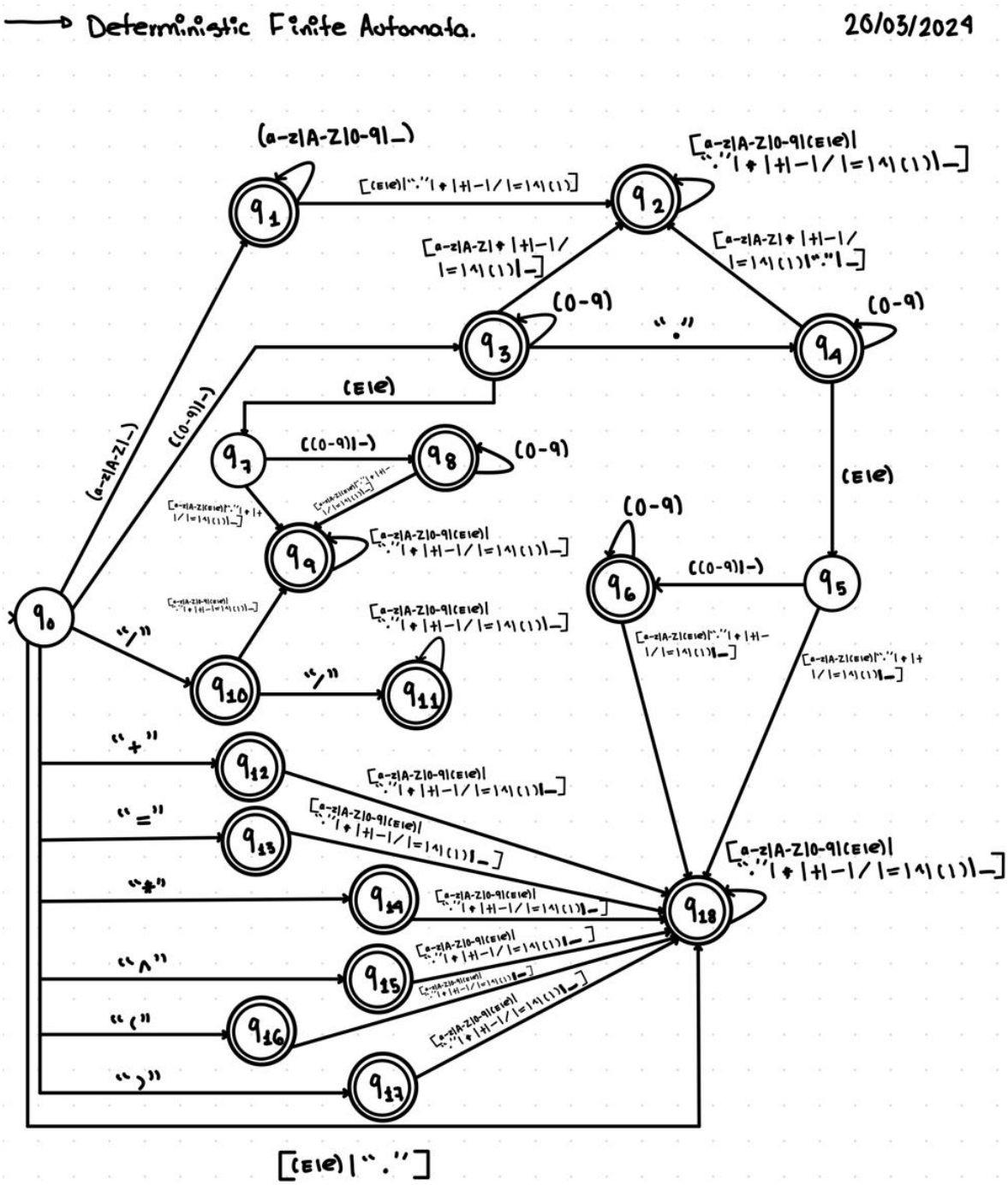


Tabla de transición del DFA:

	(a-z)	(A-Z)	"_"	(0-9)	"."	(E|e)	"/"	"="	"-"	"+"	"*"	"^"	"("	")"	
>q0	q1	q1	q1	q3	q18	q18	q10	q13	q3	q12	q14	q15	q16	q17	
*q1	q1	q1	q1	q1	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	
*q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	q2	Dead State
*q3	q2	q2	q2	q3	q3	q3	q2	q2	q2	q2	q2	q2	q2	q2	
*q4	q2	q2	q2	q4	q2	q5	q2	q2	q2	q2	q2	q2	q2	q2	
q5	q18	q18	q18	q6	q18	q18	q18	q18	q6	q18	q18	q18	q18	q18	
*q6	q18	q18	q18	q6	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
q7	q9	q9	q9	q8	q9	q9	q9	q9	q8	q9	q9	q9	q9	q9	
*q8	q9	q9	q9	q8	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	
*q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	q9	Dead State
*q10	q9	q9	q9	q9	q9	q9	q11	q9	q9	q9	q9	q9	q9	q9	
*q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	q11	
*q12	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q13	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q14	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q15	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q16	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q17	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	
*q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	q18	Dead State

Descripción de conjuntos de estados:

Estados muertos: {q2, q9, q18}

Variables: {q0, q1}

Enteros y reales: {q0, q3, q4, q5, q6, q7, q8}

Comentarios: {q10, q11}

Operador “+”: {q0, q12}

Operador “-”: {q0, q3}

Operador “=”: {q0, q13}

Operador “*”: {q0, q14}

Operador “^”: {q0, q15}

Signo “(”: {q0, q16}

Signo “)”: {q0, q17}