Cristian Pérez C16011

Documentación Proyecto 1

Diseño de la aplicación:

Se utilizaron clases para:

- Árbol sintáctico
- NFA
- DFA
- DFA Directo

La mayoría tenía un campo de data y de vecinos donde se indican las transiciones para el estado específico. El AFD guarda el AFN respectivo para el estado es decir que estado A por ejemplo tiene un AFN [‘q1’, ‘q11’,….].

Utils:

- Varias funciones para reutilizar, más que todo para procesar la expresión como sustituir los operadores +,?, por ejemplo.

Main:

- Se corre acá el programa y tiene algunas funciones que involucran a todas las clases como procesar todas las operaciones del árbol y guardarlas hacia el afn, armar el afn y dfa

Discusión:

Decidí retirar por completo el dfa directo, armé el primero intuitivamente y empezó a dar errores con pruebas posteriores, no llegaba siquiera a decir si aceptaba o no. Luego eliminé gran parte del código porque cometí el error de pensar que el dfa iba a tener 1 solo estado de aceptación. La 2da expresión de la hoja de trabajo funcionaba en este punto pero pensé que no funcionaba por no contemplar lo del dfa y lo borré porque explotó la expresión anterior. Posteriormente, probando a|b fue que me di cuenta de que el afd estaba bien y cambie para que revisara correctamente los estados de aceptación.

Realizar una prueba con a** también fue útil ya que aunque el afn se construye correctamente la revisión de aceptación no era correcta, por lo que en vez de revisar un estado final inexistente, simplemente se revisa si la palabra que se ingresa es nada, entonces revisar si dentro de los estados iniciales hay uno de aceptación.

Ejemplos y pruebas realizadas:

Prueba 1

```
expresion regular:  (a*|b*)c
ingrese la palabra a probar: aaaaaabc
esta palabra NO la puede formar el lenguaje, nfa
tiempo nfa:  0.0009627342224121094
esta palabra NO la puede formar el lenguaje, dfa
tiempo dfa:  0.0
esta palabra NO la puede formar el lenguaje, ddfa
tiempo dfa directo:  0.0

D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>
```

```
expresion regular:  (a*|b*)c
ingrese la palabra a probar: caaaaab
esta palabra NO la puede formar el lenguaje, nfa
tiempo nfa:  0.0
esta palabra NO la puede formar el lenguaje, dfa
tiempo dfa:  0.0009982585906982422
esta palabra NO la puede formar el lenguaje, ddfa
tiempo dfa directo:  0.0

D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>
```

```
expresion regular:  (a*|b*)c
ingrese la palabra a probar: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac
esta palabra SI la puede formar el lenguaje, nfa
tiempo nfa:  0.03091573715209961
esta palabra SI la puede formar el lenguaje, dfa
tiempo dfa:  0.0
esta palabra SI la puede formar el lenguaje, ddfa
tiempo dfa directo:  0.0009965896606445312

D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>
```

AFN

ESTADOS: ['q8', 'q6', 'q2', 'q0', 'q1', 'q3', 'q4', 'q5', 'q7', 'q9', 'q10']

SIMBOLOS: ['a', 'b', 'c']

INICIO: q8

ACEPTACION: q10

TRANSICION ["q8: {'\\x00': ['q2', 'q6']}", "q6: {'\\x00': ['q4', 'q7']}", "q2: {'\\x00': ['q0', 'q3']}", "q0: {'a': ['q1']}", "q1: {'\\x00': ['q0', 'q3']}", "q3: {'\\x00': ['q9']}", "q4: {'b': ['q5']}", "q5: {'\\x00': ['q4', 'q7']}", "q7: {'\\x00': ['q9']}", "q9: {'c': ['q10']}", 'q10: {}']

DFA

ESTADOS: ['A', 'B', 'C', 'D']

SIMBOLOS: ['a', 'b', 'c']

INICIO: A

ACEPTACION: D

TRANSICION ["A: {'a': ['q0', 'q1', 'q3', 'q9'], 'b': ['q4', 'q5', 'q7', 'q9'], 'c': ['q10']}", "B: {'a': ['q0', 'q1', 'q3', 'q9'], 'b': [], 'c': ['q10']}", "C: {'a': [], 'b': ['q4', 'q5', 'q7', 'q9'], 'c': ['q10']}", 'D: {}']

Prueba 2

```
ingrese la palabra a probar: b
['q5', 'q3']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
ingrese la palabra a probar: a
['q5', 'q1']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>python main.py
op concat ['(', 'a', '|', 'b', ')']
[['a', '|', 'b']]
['q2', 'q0']
expresion:  (a|b)
ingrese la palabra a probar: ab
[]
esta palabra no la puede formar el lenguaje, nfa
esta palabra no la puede formar el lenguaje, dfa
```

AFN

ESTADOS['q4', 'q0', 'q1', 'q2', 'q3', 'q5']

SIMBOLOS['a', 'b']

INICIO q4

ACEPTACION q5

TRANSICION["q4: {'\\x00': ['q0', 'q2']}", "q0: {'a': ['q1']}", "q1: {'\\x00': ['q5']}", "q2: {'b': ['q3']}", "q3: {'\\x00': ['q5']}", 'q5: {}']

DFA

ESTADOS['A', 'B', 'C']

SIMBOLOS['a', 'b']

INICIO A

ACEPTACION['B', 'C']

TRANSICION["A: {'a': ['q1', 'q5'], 'b': ['q3', 'q5']}", "B: {'a': [], 'b': []}", "C: {'a': [], 'b': []}"]

Dos estados de aceptación

Prueba 3

```
D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>python main.py
ingrese la expresion: a|b|c
op concat ['a', '|', 'b', '|', 'c']
le tree
[['a', '|', 'b'], ['|', 'c']]
['q2', 'q0']
expresion:  a|b|c
ingrese la palabra a probar: a
['q5', 'q8', 'q1']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa

D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>
```

```
ingrese la palabra a probar: c
['q8', 'q7']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
D:\Documents\uvg 2020\diseño de lenguajes\proyectos\proyecto1\final>python main.py
ingrese la expresion: a|b|c
op concat ['a', '|', 'b', '|', 'c']
le tree
[['a', '|', 'b'], ['|', 'c']]
['q0', 'q2']
expresion:  a|b|c
ingrese la palabra a probar: b
['q8', 'q5', 'q3']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

AFN

ESTADOS['q8', 'q6', 'q2', 'q0', 'q1', 'q3', 'q4', 'q5', 'q7', 'q9', 'q10']

SIMBOLOS['a', 'b', 'c']

INICIO q8

ACEPTACION q10

TRANSICION["q8: {'\\x00': ['q2', 'q6']}", "q6: {'\\x00': ['q4', 'q7']}", "q2: {'\\x00': ['q0', 'q3']}", "q0: {'a': ['q1']}", "q1: {'\\x00': ['q0', 'q3']}", "q3: {'\\x00': ['q9']}", "q4: {'b': ['q5']}", "q5: {'\\x00': ['q4', 'q7']}", "q7: {'\\x00': ['q9']}", "q9: {'c': ['q10']}", 'q10: {}']

DFA

ESTADOS['A', 'B', 'C', 'D']

SIMBOLOS['a', 'b', 'c']

INICIO A

ACEPTACION['D']

TRANSICION["A: {'a': ['q0', 'q1', 'q3', 'q9'], 'b': ['q4', 'q5', 'q7', 'q9'], 'c': ['q10']}", "B: {'a': ['q0', 'q1', 'q3', 'q9'], 'b': [], 'c': ['q10']}", "C: {'a': [], 'b': ['q4', 'q5', 'q7', 'q9'], 'c': ['q10']}", 'D: {}']

Or sobre or regresa a 1 estado de aceptación

Prueba 4

```
expresion:  b*ab?
ingrese la palabra a probar: bbbbbbbbbbbbbab
['q10', 'q6']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
expresion:  b*ab?
ingrese la palabra a probar: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbba
['q7', 'q8', 'q9', 'q5', 'q10', 'q4']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
expresion:  b*ab?
ingrese la palabra a probar: babb
[]
esta palabra no la puede formar el lenguaje, nfa
esta palabra no la puede formar el lenguaje, dfa
```

AFN

ESTADOS['q2', 'q0', 'q1', 'q3', 'q4', 'q9', 'q5', 'q6', 'q7', 'q8', 'q10']

SIMBOLOS['a', 'b']

INICIO q2

ACEPTACION q10

TRANSICION["q2: {'\\x00': ['q0', 'q3']}", "q0: {'b': ['q1']}", "q1: {'\\x00': ['q0', 'q3']}", "q3: {'a': ['q4']}", "q4: {'\\x00': ['q9']}", "q9: {'\\x00': ['q5', 'q7']}", "q5: {'b': ['q6']}", "q6: {'\\x00': ['q10']}", "q7: {'\\x00': ['q8']}", "q8: {'\\x00': ['q10']}", 'q10: {}']

DFA

ESTADOS['A', 'B', 'C', 'D']

SIMBOLOS['a', 'b']

INICIO A

ACEPTACION['B', 'D']

TRANSICION["A: {'a': ['q10', 'q4', 'q5', 'q7', 'q8', 'q9'], 'b': ['q0', 'q1', 'q3']}", "B: {'a': [], 'b': ['q10', 'q6']}", "C: {'a': ['q10', 'q4', 'q5', 'q7', 'q8', 'q9'], 'b': ['q0', 'q1', 'q3']}", 'D: {}']

Prueba 5

```
expresion:  a**
ingrese la palabra a probar: a
['q5', 'q0', 'q3', 'q1']
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
expresion:  a**
ingrese la palabra a probar: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
['q0', 'q3', 'q5', <Afn.Node object at 0x000002084CA0E5C8>]
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

```
expresion:  a**
ingrese la palabra a probar:
esta palabra si la puede formar el lenguaje, nfa
esta palabra si la puede formar el lenguaje, dfa
```

Vacío