

Reporte: Práctica 2

Barrera Pérez Carlos Tonatihu
Profesor: Saucedo Delgado Rafael Norman
Compiladores
Grupo: 3CM6

2 de septiembre de 2017

Índice

1. Introducción	2
2. Desarrollo	3
3. Resultados	3
4. Conclusiones	4
Referencias	4

1. Introducción

Existen diferentes formas para poder construir un automata finito no determinista a partir de una expresión regular, en esta práctica se utilizo la construccion de Thomson [1] con este algoritmo cada vez que se representa algun simbolo, una union, concatenacion, cerradura de Kleene o positiva se crean dos nuevos estados. El algoritmo consiste en lo siguiente:

- Para ϵ se construyen dos estados y la transicion entre ellos se etiqueta con epsilon como en la figura 1

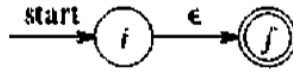


Figura 1: Epsilon representado en un automata.

- Para algun simbolo del alfabeto se realiza lo mismo que para ϵ pero esta vez se etiqueta con el respectivo simbolo como el ejemplo de la figura 2.

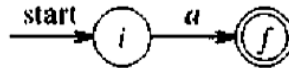


Figura 2: Representación de la transicion de un estado a otro atravez del simbolo a .

- La union de dos expresiones regulares s y t se representa a travez del uso de transiciones epsilon como se muestra en la figura 3.

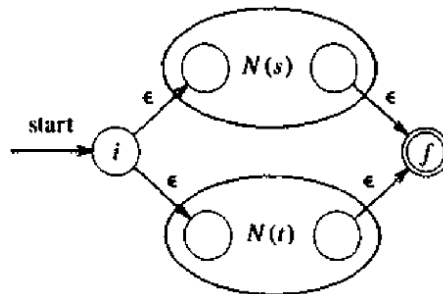


Figura 3: Union de dos expresiones regulares $(s|t)$ representadas en un AFN- ϵ .

- La concatenación de dos expresiones regulares s y t se realiza tomando el estado inicial de la transición t y convirtiendolo en el estado final de la transición producida por la expresión regular s como en la figura 4.

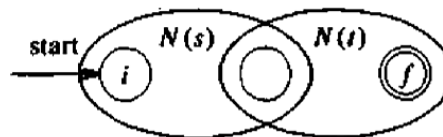


Figura 4: Concatenación (st) mediante la fusion de un estado final y uno inicial respectivamente.

- La cerradura de Kleene de una expresión regular s^* se realiza usando transiciones epsilon entre los estados que componen esta expresión. Esto se muestra en la figura 5.

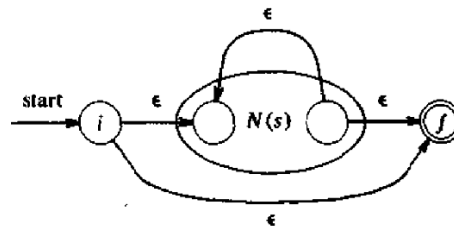


Figura 5: Cerradura (s^*).

- La cerradura positiva es similar a la de Kleene pero esta no realiza una transición del estado inicial al final.

Finalmente, para la evaluación de la expresión regular primero la pasamos a postfijo con el algoritmo shunting yard [2] que se utiliza para la evaluación de expresiones matemáticas pero adaptándolo a expresiones regulares.

2. Desarrollo

hola

3. Resultados

A continuación se presenta el resultado de la implementación de la clase *Analizador* en donde se convierte una expresión regular a un AFN para después evaluar algunas cadenas y ver si son válidas. Las pruebas se realizaron con el siguiente código, el cual ocupa la clase *AFN* que se contruye gracias a la clase *Analizador* para poder llevar a cabo la evaluación de algunas cadenas.

```

tona@anti: ~/Documents/quinto/compiladores/Practicas
File Edit View Search Terminal Help
+ Practicas git:(master) x python iniciar.py
Generando el automata de la expresion: (a|b)*abb ...
['a', 'b', '|', '*', 'a', 'b', 'b', '.', '.', '.']
ESTADO
ESTADO
UNION
CERRADURA KLEENE
ESTADO
ESTADO
ESTADO
CONCATENACION
CONCATENACION
CONCATENACION
Inicial: 7 Finales: {14}
Transiciones:
1->2: a
3->4: b
5->3: e
5->1: e
4->6: e
2->6: e
7->5: e
6->5: e
6->9: e
7->9: e
9->11: a
11->13: b
13->14: b
Pruebas sobre el automata generado...
-Ingresa una cadena: ba

```

Figura 6: Transormando la expresión regular $(a|b)^*abb$ a su respectivo automata.

Como se puede observar en la figura 6 la expresion regular se pasa a notación posfija y después se puede observar como se realiza la creación de las transiciones, lo siguiente que aparece es mostrar los estados inicial y final y finalmente se muestran las transiciones resultantes.

Después se realizaron pruebas sobre el automata que se genero de forma similar a la práctica anterior.

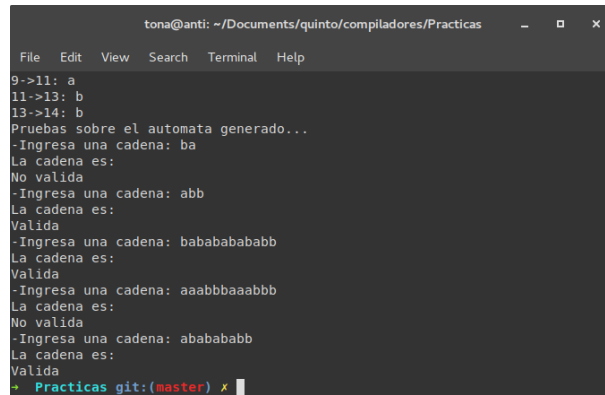
A screenshot of a terminal window titled 'tona@anti: ~/Documents/quinto/compiladores/Practicas'. The terminal shows a series of commands and outputs for testing a generated automaton. The commands are: '9->11: a', '11->13: b', and '13->14: b'. The output shows 'Pruebas sobre el automata generado...' followed by several test cases. Each test case starts with '-Ingresa una cadena: [string]' and is followed by 'La cadena es:' and either 'Valida' or 'No valida'. The test cases are: 'ba' (Valida), 'abb' (Valida), 'bababababab' (Valida), 'aaabbbbaabbb' (No valida), and 'abababab' (Valida). The terminal ends with a prompt 'Practicas git:(master) x'.

Figura 7: Pruebas sobre el automata generado.

La generación del automata se realizo de forma correcta (figura 7), ya que la evaluación de las cadenas fue satisfactoria esto quiere decir que las transiciones que se muestran en la figura 6 no tuvieron errores.

4. Conclusiones

La elaboración de esta práctica fue un poco más sencilla que la anterior debido a que solo consistio en implementar el algoritmo de Thomson y con ello poder construir cualquier AFN, la parte complicada de modelar este algoritmo fue la concatenación ya que requirio unos pasos extra a diferencia del resto de construcciones.

Y como se pudo observar en las pruebas la conversion de expresion regular a automata se realiza correctamente. Esto demuestra la versatilidad que tienen las expresiones regulares y los automatas además de que sera de mucha utilidad cuando realicemos nuestro analizador lexico que su principal tarea es revisar el vocabulario de nuestro compilador.

Referencias

- [1] V. A. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 1st ed., 1986.
- [2] P. Oser, "The shunting yard algorithm." <http://www.oxfordmathcenter.com/drupal17/node/628>. Consultado: 2017-09-2.