

Implementación en Java de un programa el cuál recibe una ER y la procesa convirtiéndola en Notación Postfija seguido de AFN- ϵ para concluir en AFN al cuál se le calculan todas las cadenas aceptadas por este.

Matemáticas Computacionales

Automatas y ER's



**TECNOLÓGICO
DE MONTERREY®**

Luis Eduardo Gutiérrez Pedroza A01225387
César de la Torre A0
Antonio Reyes Espinoza A0

Proyecto Final – Matemáticas Computacionales

I. Análisis del problema

a) Descripción del Modulo 1

El Programa recibe una Expresión Regular la cual convierte a Notación Postfija

b) Descripción del Modulo 2

Se recibe una Notación postfija, la cual se procesa y crea un Autómata Finita no Determinista con transiciones épsilon, con los estados dados en consecuencia a la notación.

c) Descripción del Modulo 3

El Modulo 3 recibe un Autómata Finito No Determinista con transiciones épsilon y da cómo salida un AFN sin transiciones épsilon, el cuál se crea mediante el proceso de conversión de un AFN- ϵ a AFD, pero dejando afuera el estado de limbo que se agrega.

d) Descripción del Modulo 4

Recibe un AFN creado previamente en el modulo 3 y en base a este calcula todas las cadenas que cumplen con la expresión regular dada al inicio, la cual fue recreada en los diferentes autómatas.

II. Diseño de la solución

a. Modulo 1

- Recibo una expresión regular, con la que introduce símbolos a una pila y las va sacando dependiendo de la prioridad del operador, para así crear una expresión postfija la cual puede ser más fácil convertida por una computadora.

b. Modulo 2

- Recibe una notación postfija, conforme la va leyendo crea estados para los caracteres y aplicando los operadores cuando es debido utilizando un arreglo que indica cuales son los últimos estados iniciales y finales de diversos autómatas sobre los cuales opera.

c. Modulo 3

- El modulo 3 recibe un AFN con transiciones épsilon y después calcula las transiciones de un solo carácter del alfabeto en un estado con su clausura, para agregarlo a la lista de estados del nuevo AFN, después sigue haciendo esto para cada uno de los estados y caracteres agregando de los nuevos estados no existentes, al mismo tiempo escribe una nueva tabla parecida a la de un AFD, pero con transiciones faltantes, las cuales llevarían al limbo, estas se rellenan con un -1. Al terminar este proceso toma el AFN-e

existente y filtra sólo los caracteres utilizados, para crear así la nueva tabla del AFN.

d. Modulo 4

- Utiliza los 3 módulos anteriores para después leer línea por línea un texto y devolver sin repeticiones, las cadenas aceptadas.

III. Pruebas

a. Prueba Modulo 1

Entrada:

(gato,perro)+.*(auto,bote)+

Salida:

ga#t#o#pe#r#r#o# ,+ . *#au#t#o#bo#t#e# ,+#

b. Prueba Modulo 2

La prueba del modulo dos es muy extensa ya que incluye estados para cada letra del alfabeto y hasta el modulo 3 es depurado, a continuación una prueba con una cadena corta

ER = (a,b)*

Notación Postfija = ab,*

Numero de Estados: 8

Estado inicial: q6 Estado Final: q7

Estado q0 simbolo : a estados: [1]

Estado q1 simbolo : ε estados: [5]

Estado q2 simbolo : b estados: [3]

Estado q3 simbolo : ε estados: [5]

Estado q4 simbolo : ε estados: [0, 2] simbolo : ε estados: [0, 2]

Estado q5 simbolo : ε estados: [7, 4] simbolo : ε estados: [7, 4]

Estado q6 simbolo : ε estados: [4, 7] simbolo : ε estados: [4, 7]

Estado q7

AFN final

[][a][b]

[0][1][2]

[1][1][2]

[2][1][2]

c. Prueba Modulo 3



Salida:

[][a][b][c]
 [0][1][][2]
 [1][1][3][2]
 [2][1][][3]
 [3][1][][]

d. Prueba Modulo 4

Con este texto:

los gatos me gustan mucho por que los gatos son lindos
 a mi me gustan los lindos gatos que son peluditos
 gato perro gato pollo gato
 gatoperrogatopollo
 hola me gustan los gatos pero no los pollos

Y la ER: gato,perro

El programa regresa las dos cadenas

gato

perro