**Práctica 3:**

Implementación del algoritmo para convertir un AFN en AFD

**Alumno:**

Meza Vargas Brandon David

**Grupo:**

3CM13

**Profesor:**

Sánchez Juárez José

**Índice**

[**Objetivo** 3](#_Toc120302491)

[**Cuestionario** 3](#_Toc120302492)

[**Conclusiones** 17](#_Toc120302493)

# **Objetivo**

Implementar un procedimiento para convertir un AFN a un AFD

# **Cuestionario**

**1.- Entregar un procedimiento para convertir el AFN en AFD aplicando el algoritmo de subconjuntos**

Para pasar un AFN a un AFD con el algoritmo de subconjuntos tenemos que usar dos operaciones:

* Operación cerradura épsilon: cerr-eps()
* Operación movimiento que se aplica a todo el alfabeto: mov(Estado, símbolo).

El procedimiento para convertir de un AFN a AFD es sencillo pues solo requerimos dos pasos.

1. Aplicar la operación cer-eps() al estado inicial del autómata AFN
2. Aplicar la operación movimiento para todos lo símbolos del alfabeto al estado obtenido de aplicar la operación cerradura épsilon al estado inicial del AFN. Esta operación se aplica hasta cuando ya no existe un estado nuevo.

Para que este procedimiento quede más claro veamos un ejemplo. Pasando el AFN de un identificador a su AFD.

Diagrama, Forma

Descripción generada automáticamente

Ilustración . AFN de un identificador

El primer paso es aplicar la operación cerr-eps al estado inicial, y a este conjunto lo etiquetaremos como q0.

**Cerr-eps({0}) = {0} = q0**

Ahora aplicaremos la operación movimiento considerando el siguiente alfabeto

**Σ = {I, d}**

Teniendo nuestro alfabeto hacemos la operación al estado q0.

**Mov(q0, I) = {1}**

A este conjunto de estados se le aplica la operación cerr-eps(q0, 1), de esta manera tenemos:

**Cerr-eps(mov(q0,I)) = cerr-eps({1}) = {1, 2, 3, 4, 6, 9} = q1**

Ahora lo haremos para el otro estado:

**Cerr-eps(mov(q0,d)) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q1

**Cerr-eps(mov(q1,I)) = cerr-eps{5} = {5, 8, 3, 4, 6, 9} = q2**

Ahora con el símbolo d

**Cerr-eps(mov(q1,d)) = cerr-eps({7}) = {7, 8, 3, 4, 6, 9} = q3**

Ahora lo haremos con q2

**Cerr-eps(mov(q2,I)) = cerr-eps({5}) = {5, 8, 3, 4, 6, 9} = q2**

Para el simbolo d:

**Cerr-eps(mov(q2,d)) = cerr-eps({7}) = {7, 8, 3, 4, 6, 9} = q3**

Ahora para el estado q3

**Cerr-eps(mov(q3,I)) = cerr-eps({5}) = {5, 8, 3, 4, 6, 9} = q2**

Ahora para el símbolo d:

**Cerr-eps(mov(q3,d)) = cerr-eps({7}) = {7, 8, 3, 4, 6, 9} = q3**

Como no hay estados nuevos hemos finalizado, ya solo tenemos que crear nuestro AFD con los estados resultantes, resultando en el siguiente AFD.

Diagrama

Descripción generada automáticamente

Ilustración . AFD

**2.- Entregar un procedimiento para convertir el AFN en AFD aplicando el árbol de análisis sintáctico**

Este procedimiento se utiliza para pasar de una ER a un AFD mínimo. Aquí tenemos que considerar los siguientes tipos de nodos.

Texto

Descripción generada automáticamente con confianza baja

Ilustración . Tipos de nodos

De igual forma tenemos que considerar las siguientes reglas para la aplicación de las funciones.

**Función anulable().** Es **true** para un nodo **n** de un árbol de análisis sintáctico y si y solamente si la subexpresión representada por **n** tiene ε en su lenguaje. Esto es, la subexpresión se puede hacer nula o cadena vacia, aunque hay muchas otras cadenas que pueden representarse también. Es necesario conocer qué nodos son las raı́ces de las subexpresiones que generan lenguajes que incluyen la cadena vacı́a. A dichos nodos se les denomina anulables, y la función anulable(n) se define como **true** si el nodo **n** es anulable, y **false** en caso contrario.

**Etiquetación de las hojas n.** Si la hoja **n** esta etiquetada con el caracter vacio ε, la función **anulable(n)** es **true**. O si la hoja **n** esta etiquetada con la posición **i** entonces la función **anulable(n)** es **false**. La función anulable también se aplica a las operaciones de: Concatenación, Alternativa y cerradura.

**La función anulable(n) para la concatenación (nodo-cat).** Se aplica a cada una de las alternativas la función anulable(a) y anulable(b). **La función primerapos().** Es **if** anulable(n) **then** primerapos(n1) U primerapos(n2) **else** primerapos(n1). **La función ultimapos().** Es **if** anulable(n) **then** ultimapos(n1) U ultimapos(n2) **else** ultimapospos(n1).

**La función anulable(n) para la alternativa (nodo-o).** Se aplica a cada una de las alternativas la función anulable(). Y se aplica anulable(a) or anulable(b). **La función primerapos().** Es primerapos(a) U primerapos(b). **La fución ultimapos().** Es ultimapos(a) U ultimapos(b).

**La función anulable(n) para la cerradura de Kleene (nodo-ast).** Se aplica de manera directa ya que para la cerradura es **true**. **La función primerapos().** Es primerapos(n). **La función ultimapos().** Es ultimapos(n).

**La función anulable(n) para la cerradura positiva (nodo-posi).** Se aplica de manera directa a su hijo que puede ser **true** o **false**. **La función primerapos().** Es primerapos(n). **La función ultimapos().** Es ultimapos(n).

**La función anulable(n) para la interrogación (nodo-inte).** Se aplica de manera directa es **true**. **La función primerapos().** Es primerapos(n). **La función ultimapos().** Es ultimapos(n).

**Función primerapos().** La que proporciona el conjunto de posiciones que pueden concordar con el primer sı́mbolo de una cadena generada por la subexpresión con raı́z en n.

**Función siguientepos().** Es el conjunto de posiciones j tales que hay alguna cadena de entrada cd tal que i corresponde a la aparición de c y j a la aparición de d.

**Función últimapos().** La que proporciona el conjunto de posiciones que pueden concordar con el último sı́mbolo en esa cadena.

**Árbol de análisis sintáctico.** Este árbol presenta en cada uno de sus nodos el sı́mbolo no terminal, de tal manera que se presenta la sintaxis de la gramática.

Considerando lo anterior el procedimiento de este algoritmo es el siguiente.

1. Aumentar la expresión regular escribiendo el símbolo # al final de la ER
2. Se terminan los tipos de nodos que contiene la expresión regular
3. Se construye su árbol de análisis enumerando los nodos hoja de izquierda a derecha.
4. Se calculan las funciones detalladas anteriormente, para esto nos podemos ayudar de la siguiente tabla.

Tabla

Descripción generada automáticamente con confianza media

1. Los conjuntos obtenidos de primerapos() se escribem a la izquierda del nodo, los conjuntos obtenidos de ultimapos() se escriben a la derecha del nodo
2. Aplicar la operación transición con todo el vocabulario
3. Se crea el AFD a partir de la tabla generada con las operación transición.

**2.- Aplicar los procedimientos de las preguntas 1 y 2 al siguiente código de entrada para obtener un AFD**

**if(a < b)**

**return a + b;**

**else**

**return a - b;**

Para la realización del ejercicio se tomará la expresión regular **if(a<b)** a partir de esta obtendremos su AFN y con los algoritmos de las preguntas 1 y 2 obtendremos el AFD

**Obteniendo el AFN**

Diagrama

Descripción generada automáticamente

Ilustración . AFN

**Algoritmo de subconjuntos**

El primer paso es aplicar la operación cerr-eps al estado inicial, y a este conjunto lo etiquetaremos como q0.

**Cerr-eps({0}) = {0} = q0**

Ahora aplicaremos la operación movimiento considerando el siguiente alfabeto

**Σ = {i,f,(,a,<,b,)}**

Teniendo nuestro alfabeto hacemos la operación al estado q0.

**Cerr-eps(mov(q0,i)) = cerr-eps({1}) = {2} = q1**

**Cerr-eps(mov(q0,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q0,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q0,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q0,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q0,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q0,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q1 y el alfabeto

**Cerr-eps(mov(q1,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q1,f)) = cerr-eps{**3**} = {3,** 4**} = q2**

**Cerr-eps(mov(q1,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q1,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q1,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q1,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q1,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q2 y el alfabeto

**Cerr-eps(mov(q2,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q2,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q2,()) = cerr-eps{**5**} = {**5, 6**} = q3**

**Cerr-eps(mov(q2,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q2,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q2,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q2,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q3 y el alfabeto

**Cerr-eps(mov(q3,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q3,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q3,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q3,a)) = cerr-eps{**7**} = {**7, 8**} = q4**

**Cerr-eps(mov(q3,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q3,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q3,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q4 y el alfabeto

**Cerr-eps(mov(q4,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q4,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q4,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q4,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q4,<)) = cerr-eps{**9**} = {**9,10**} = q5**

**Cerr-eps(mov(q4,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q4,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q5 y el alfabeto

**Cerr-eps(mov(q5,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q5,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q5,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q5,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q5,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q5,b)) = cerr-eps{**11**} = {**11, 12**} = q6**

**Cerr-eps(mov(q5,))) = cerr-eps{**ø**} = {**ø**}**

Ahora aplicaremos las mismas operaciones para el nuevo estado q6 y el alfabeto

**Cerr-eps(mov(q6,i)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,f)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,()) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,a)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,<)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,b)) = cerr-eps{**ø**} = {**ø**}**

**Cerr-eps(mov(q6,))) = cerr-eps{**13**} = {**13**} = q7**

Formando el AFD queda.

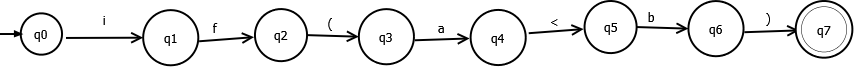


Ilustración . AFD

**Método del arbol**

Primero aumentamos la ER

**If(a<b)#**

Creando árbol

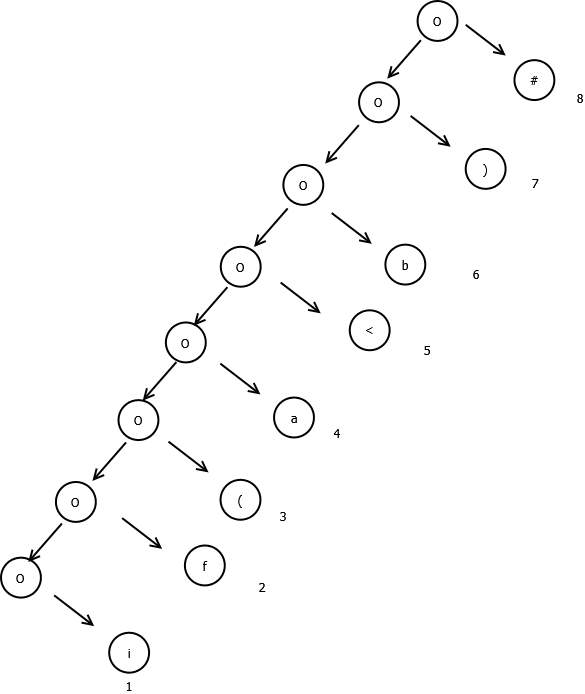


Ilustración . Árbol

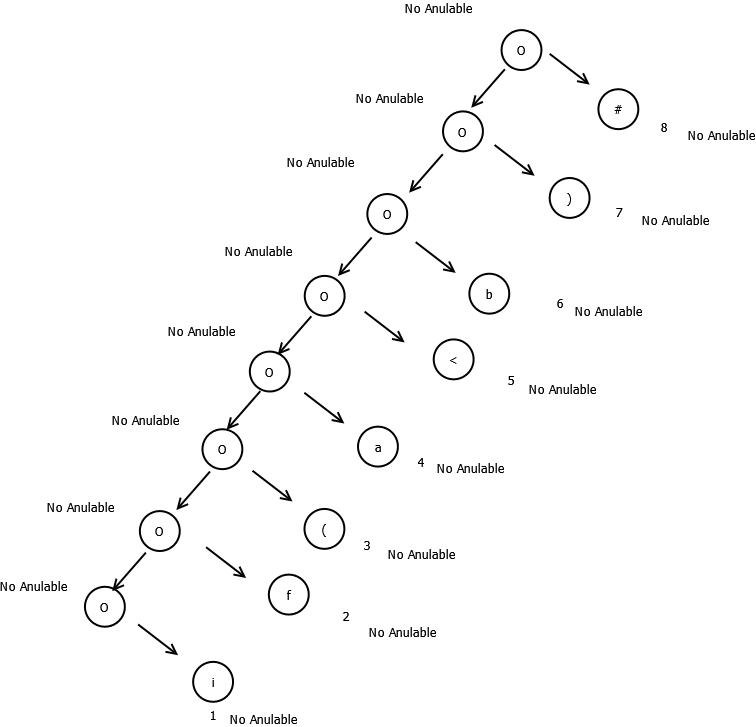


Ilustración . Anulable()

Calculando primera y última posición

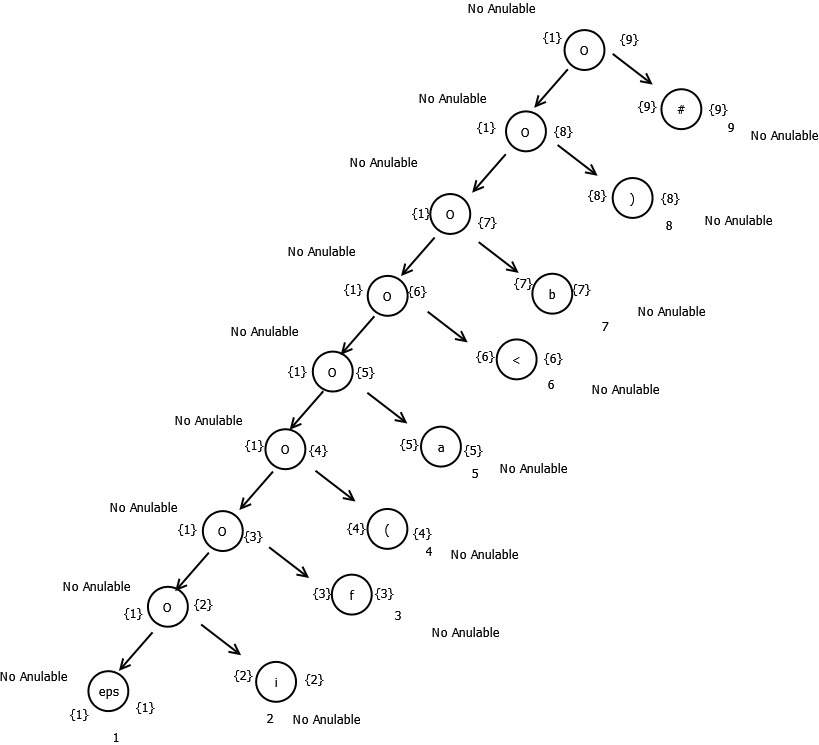
****

Ilustración . Calculando primera y última posición

Obteniendo siguiente posición

|  |  |
| --- | --- |
| Nodo i | Siguiente pos |
| 1 | {2} |
| 2 | {3} |
| 3 | {4} |
| 4 | {5} |
| 5 | {6} |
| 6 | {7} |
| 7 | {8} |
| 8 | {9} |
| 9 | {10} |
| 10 | - |

Calculando estados

**Primerapos(raíz) = {2} = q0**

Obteniendo tran y tranD de todo el vocabulario

**Tran[q0,i] = {2}**

**tranD[q0,i] = siguientepos(2) = {3} = q1**

**Tran[q0,f] = {** ø **}**

**Tran[q0,(] = {**ø**}**

**Tran[q0,a] = {**ø**}**

**Tran[q0,<] = {**ø**}**

**Tran[q0,b] = {**ø**}**

**Tran[q0,)] = {**ø**}**

Con q1:

**Tran[q1,i] = {**ø**}**

**Tran[q1,f] = {**3**}**

**tranD[q1,f] = siguientepos(3) = {4} = q2**

**Tran[q1,(] = {**4**}**

**Tran[q1,a] = {**ø**}**

**Tran[q1,<] = {**ø**}**

**Tran[q1,b] = {**ø**}**

**Tran[q1,)] = {**ø**}**

Con q2:

**Tran[q2,i] = {**ø**}**

**Tran[q2,f] = {**ø**}**

**Tran[q2,(] = {** ø **}**

**tranD[q2,(] = siguientepos(4) = {5} = q3**

**Tran[q2,a] = {** ø **}**

**Tran[q2,<] = {**ø**}**

**Tran[q2,b] = {**ø**}**

**Tran[q2,)] = {**ø**}**

Con q3:

**Tran[q3,i] = {**ø**}**

**Tran[q3,f] = {**ø**}**

**Tran[q3,(] = {** ø **}**

**Tran[q3,a] = {** 5 **}**

**tranD[q3,a] = siguientepos(5) = {6} = q4**

**Tran[q3,<] = {** ø **}**

**Tran[q3,b] = {**ø**}**

**Tran[q3,)] = {**ø**}**

Con q4:

**Tran[q4,i] = {**ø**}**

**Tran[q4,f] = {**ø**}**

**Tran[q4,(] = {** ø **}**

**Tran[q4,a] = {** ø **}**

**Tran[q4,<] = {** 6 **}**

**tranD[q4,<] = siguientepos(6) = {7} = q5**

**Tran[q4,b] = {** ø **}**

**Tran[q4,)] = {**ø**}**

Con q5:

**Tran[q5,i] = {**ø**}**

**Tran[q5,f] = {**ø**}**

**Tran[q5,(] = {** ø **}**

**Tran[q5,a] = {** ø **}**

**Tran[q5,<] = {** ø **}**

**Tran[q5,b] = {** 7 **}**

**tranD[q4,b] = siguientepos(7) = {8} = q6**

**Tran[q5,)] = {** ø **}**

Con q6:

**Tran[q7,i] = {**ø**}**

**Tran[q7,f] = {**ø**}**

**Tran[q7,(] = {** ø **}**

**Tran[q7,a] = {** ø **}**

**Tran[q7,<] = {** ø **}**

**Tran[q7,b] = {** ø **}**

**Tran[q7,)] = {** 8 **}**

**tranD[q5,)] = siguientepos(8) = {9} = q7**

Obteniendo el AFD mínimo

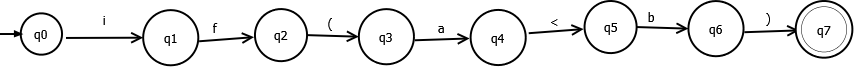


Ilustración . AFD método árbol

# **Conclusiones**

Gracias a esta práctica y el ejercicio que venía de transformar una AFN a AFD pude comprender de mejor manera como funciona este método, además de aplicar otro método que es el del árbol para convertir una ER a un AFD, debo decir que este último es más largo y un poco más complicado que el algoritmo de subconjuntos, sin embargo al final se logró resolver el objetivo de la práctica sin mayores complicaciones.