Instituto Politécnico Nacional

Escuela Superior de Cómputo

Compiladores

Practica 3. Implementación de algoritmo AFN -> AFD

Alumno (a): Brenda Guadalupe Sánchez Sánchez

Grupo: 3CV5

Profesor: M en C. Saucedo Delgado Norman Rafael

**Introducción**

En el presente trabajo se lleva a cabo la implementación del algoritmo del algoritmo de subconjuntos, esta implementación fue obtenida del libo de Compiladores pg. 152 [1]

A continuación se muestran las principales operaciones que se aplican sobre los afn con el fin de obtener un afd.

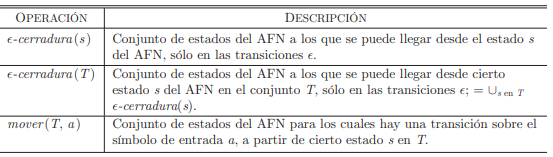


Figura 1.1 Operaciones sobre los estados del AFN [1]

Siendo un autómata aquella máquina abstracta que es capaz de resolver problemas, definiremos que es un AFN y AFD

Los AFN y AFD se caracterizan por tener las siguientes características:

1. Conjunto de estados Q
2. Alfabeto
3. Estado inicial
4. Función de transición
5. Conjunto de estados finales o de aceptación

AFN: “Consideremos el lenguaje regular A representado por c\* (a u be\*)\*. Si dada una cadena vv se nos pregunta si vv pertenece a A, debemos analizar no sólo los caracteres que aparecen en vv, sino también sus posiciones relativas.” [2]

AFD: “Para trabajar con los AFD es necesario usar ciertas definiciones y notaciones. Si M es un AFD, entonces el lenguaje aceptado por M es

L (Ai) = {vv e L\* | w es aceptada por M }

Por tanto, L (M) es el conjunto de cadenas que hacen que M pase de su estado inicial a un estado de aceptación.” [2]

**Desarrollo**

1. Contemplación de una metología

La metodología seguida en esta práctica es una de autoría propia basada el paradigma divide y vencerás con la qué es fácil visualizar la descomposición de la solución del programa en pequeños módulos que se simplifican facilitando la solución del problema. Esta metodología contempla los siguientes pasos

1. Entendimiento del algoritmo
2. Contemplación de las estructuras sugeridas para la implementación del algoritmo
3. Posibles funciones y estructuras auxiliares extra a las mencionadas en el algoritmo del libro compiladores.
4. Codificación
5. Código de implementación

|  |
| --- |
| 1. /\*Algoritmo de subconjuntos 2. Brenda Gpe.Sánchez Sánchez 3. Compiladores 4. \*/ 5. #include <bits/stdc++.h> 6. using namespace std; 7. vector<int> epsilon;//agregar los estados posibles a ir con Epsilon 8. vector <vector<int>> nuevosEstados>; // nuevos estados después del proceso 9. stack <vector> verifEstados1; 10. stack <vector> verifEstados2; 11. void epsilonCerradura(vector<int> Estados){ // vector que contienen todos los estados con transición a epsilon 12. for (int i=0;i<Estados.size();i++){ 13. if (transitionTable[Estados[i]][E]==1); 14. epsilon.push\_back(i); 15. } 16. } 17. v 18. void moverSimb(vector <int> Estados,vector<char>Simbolos){ //Se prueban los estados con cada símbolo 19. for (int i=0;i<Estados.size();i++){ 20. for (int j=0;j<Simbolos.size();j++){ 21. if(transitionTable[Estados[i]][Simbolos[j]]=1) 22. EstadoGenerado.push\_back(); 23. } 24. } 25. } 26. bool ExisteEstado(vector<int> EstadoGenerado){ //Verificación de la existencia de un estado generado 27. vector<int> EnPila; 28. if verifEstados1.empty()==true //VerifEstados1 guarda todos los estados que se van generando 29. verifEstados1.push\_back(EstadoGenerado); 30. while (!verifEstados1.empty()){ 31. if(EstadoGenerado.size()==verifEstados1.pop().size()) 32. return false; 33. EnPila=verifEstados1.top(); 34. sort(EstadoGenerado.begin(),EstadoGenerado.end()); 35. sort(EnPila.begin(), EnPila.end()); 36. for (int i = 0; i < EstadoGenerado.size(); ++i) 37. { 38. if(EstadoGenerado[i]!=EnPila[i]) 39. return false 40. } 41. verifEstados2.push\_back(EnPila.top()); //pila auxiliar para intercambiar estados dela pila1 a la pila2 42. verifEstados1.pop(); 43. } 44. while(!=verifEstados2.empty()){ //Una vez que se comprueba que no hay estados repetidos se regresan a la pila 1 45. verifEstados1.push\_back(verifEstados2.top()) 46. verifEstados2.pop(); 47. } 48. verifEstados1.push\_back(EstadoGenerado); 49. } |

3. Conclusiones

El desarrollo de esta práctica fue un tanto confusa porque no tenía bien claro en qué formato las funciones recibían el AFN así que tome como base una tabla de transición, que no contempla la transición de un estado a más de un estado con un símbolo. Si bien la idea está bien planteada no contempla algunos casos. Me siento un tato decepcionada por no saber exactamente cómo manejar los casos no contemplados. Seguiré realizando ejercicios y mejorar está práctica.

**Referencias**

[1].Alfred V. Aho, Monica S. Lam. Compiladores. Pearson. México.2008

[2].Kelley Dean. Teoría de autómatas y lenguajes formales. Prentice Hall. España.1995