UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO

Teoría de la computación

Q U E P R E S E N T A N

Néstor Hernández Castañeda

Jonathan Rojas Simón

César Alexis Estrada Palacios

Angel Villalobos Peralta

Angelo Marlon Montoya Guerra.

Jorge Ignacio Rivera Cortez

Verónica Neri Mendoza

Profesor: Dr. René Arnulfo García Hernández

TIANGUISTENCO, MÉX. DICIEMBREMBRE DE 2017

Contenido

[Notación 3](#_Toc499894326)

[Estados Alcanzables 3](#_Toc499894327)

[Ejemplo 3](#_Toc499894328)

[Solución computacional 4](#_Toc499894329)

[Pruebas 5](#_Toc499894330)

[Conversión de un autómata finito no determinista un autómata finito determinista. 7](#_Toc499894331)

[Ejemplo 8](#_Toc499894332)

[Solución computacional: 9](#_Toc499894333)

[Pruebas 10](#_Toc499894334)

[Minimización de un Autómata Finito Determinista (AFDM) 15](#_Toc499894335)

[Ejemplo 16](#_Toc499894336)

[Solución Computacional 18](#_Toc499894337)

[Pruebas 18](#_Toc499894338)

[Validación de cadenas 20](#_Toc499894339)

[Ejemplo 20](#_Toc499894340)

[Solución Computacional 21](#_Toc499894341)

|  |  |
| --- | --- |
| **Notación** | |
| Q= Conjunto de estados, (|Q| < ∞)  A= ∑ = Alfabeto  R= r= Alfabeto de la cinta  q= = Estado inicial 𝜖 Q  B= b= Símbolo de estado en blanco  F= Conjunto de estados de aceptación  T = = Función de la transición | |
| **Estados Alcanzables** | |
| **ENTRADA:** Autómata finito determinista o autómata finito no determinista.  **SALIDA**: Autómata con estados alcanzables. | |
| Dada una entrada determinar a qué estados es posible llegar a través del estado inicial. | |
| **Ejemplo** | |
| Dada una entrada determinar a qué estados es posible llegar a través del estado inicial. | |
|  | |  |  | | --- | --- | | Estados | Condición | | →S | Alcanzable | | \*B | Alcanzable | | \*C | Alcanzable | | D | No Alcanzable | |
| **Solución computacional** | |
| 1. Se construye un mapa (TreeMap) con los estados que se obtienen de los parámetros del archivo config.txt   Representación del árbol generado a través del TreeMap y HashSet   1. Se genera un arreglo de enteros del tamaño de la cantidad de estados, donde cada índice del arreglo representa un estado del autómata. Dentro de este arreglo se utilizan 3 banderas, las cuales son:   0 → Para identificar estados no alcanzables  1 → Para identificar estados alcanzables  2 → Como auxiliar (Estado Pendiente)   1. Al inicio todas las casillas del arreglo se encuentran con la bandera 0, excepto el estado inicial que se pondrá en estado 2, y se pasa a la siguiente condición   3.1- Mientras estados pendientes = verdadero   * 1. Se cambia el valor de la condición anterior a falso y se empieza a recorrer el arreglo índice por índice, hasta que encontrar una bandera 2 (estado actual), si la encuentra cambia el valor de la bandera a 1, revisa en el mapa a que estados alcanza el estado actual, asignando a sus estados alcanzables una bandera con valor 2, si esto ocurre cambia la condición 3.1 a verdadero y se repite el inciso a.  1. Una vez que se cumplió la condición de paro (que la condición 3.1 este quede en falso). Se crea un nuevo conjunto solo con los estados que son alcanzables, es decir solo con aquellos elementos que tienen la bandera 1 y se generan los archivos de salida (o definición de un nuevo autómata). | |
| **Pruebas** | |
| 1. C:\Users\UAEMex\Pictures\EA1original.jpegOriginal | |  |  | | --- | --- | | **Estado** | **Condición** | | →0 | Alcanzable | | 1 | No Alcanzable | | 2 | No Alcanzable | | 3 | No Alcanzable | | 4 | No Alcanzable | | \*5 | No Alcanzable | | 6 | Alcanzable | | 7 | No Alcanzable | | 8 | No Alcanzable | | \*9 | No Alcanzable | | 10 | No Alcanzable | | 11 | No Alcanzable | |
| * 1. Salida   C:\Users\UAEMex\Pictures\EA1Final.jpeg |
| 1. Original/Salida   C:\Users\UAEMex\Pictures\EAOriginal1.jpeg | |  |  | | --- | --- | | **Estado** | **Condición** | | →0 | Alcanzable | | 1 | Alcanzable | | 2 | Alcanzable | | \*3 | Alcanzable | |
| 1. Original   C:\Users\UAEMex\Pictures\3.jpeg | |  |  | | --- | --- | | **Estado** | **Condición** | | →0 | Alcanzable | | 1 | Alcanzable | | 2 | Alcanzable | | 3 | Alcanzable | | 4 | Alcanzable | | \*5 | Alcanzable | | 6 | Alcanzable | | 7 | No alcanzable | | \*8 | Alcanzable | | \*9 | Alcanzable | | 10 | Alcanzable | | 11 | No Alcanzable | | 14 | No Alcanzable | |
| 3.1) Salida  C:\Users\UAEMex\Pictures\3.1.jpeg |
| 1. Original / Salida   C:\Users\UAEMex\Pictures\4.11.jpeg   1. C:\Users\UAEMex\Pictures\Diagram1.jpegOriginal / Salida | |  |  | | --- | --- | | **Estado** | **Condición** | | →0 | Alcanzable | | 1 | Alcanzable | | 2 | Alcanzable | | 3 | Alcanzable | | \*4 | Alcanzable |  |  |  | | --- | --- | | **Estado** | **Condición** | | →0 | Alcanzable | | \*1 | Alcanzable | | 2 | Alcanzable | | 3 | Alcanzable | |
|  | |
| **Conversión de un autómata finito no determinista un autómata finito determinista.** | |
| **ENTRADA:** autómata finito no determinista.  **SALIDA:** Autómata finito determinista. | |
| Construir la tabla del AFND | |
| 1. Considerar a cada celda como un conjunto de estados. 2. Para cada celda de la tabla con más de un estado, agregarlos a la tabla como un nuevo estado para hacer su definición. 3. Para hacer la definición del nuevo estado con el símbolo de transición, hacer la unión de todos los conjuntos de los estados involucrados en el nuevo estado. 4. Realizar el paso 3 hasta que todos los estados estén definidos. 5. Si un estado final forma parte de un nuevo estado, entonces el nuevo estado también es final. 6. Eliminar los estados no alcanzables desde el estado inicial. | |
| **Ejemplo** | |
| 1. Considerar a cada celda como un conjunto de estados.  |  |  | | --- | --- | | Estados | a | | →0 | { 0,1,2,3} | | 1 | 2 | | 2 | 3 | | \*3 | - |  1. Para cada celda de la tabla con más de un estado, agregarlos a la tabla como un nuevo estado para hacer su definición.  |  |  | | --- | --- | | Estados | a | | →0 | { 0,1,2,3} | | 1 | 2 | | 2 | 3 | | \*3 | - | | { 0,1,2,3} |  |  1. Para hacer la definición del nuevo estado con el símbolo de entrada 𝛼, hacer la unión de todos los conjuntos de los estados involucrados en el nuevo estado.  |  |  | | --- | --- | | Estados | a | | →0 | { 0,1,2,3} | | 1 | 2 | | 2 | 3 | | \*3 | - | | { 0,1,2,3} | { 0,1,2,3} U {2} U {3} U {-} = {0,1,2,3} |  1. Realizar el paso 3 hasta que todos los estados estén definidos. 2. Si un estado final forma parte de un nuevo estado, entonces el nuevo estado también es final.  |  |  | | --- | --- | | Estados | a | | →0 | { 0,1,2,3} | | 1 | 2 | | 2 | 3 | | \*3 | - | | { 0,1,2,3} | { 0,1,2,3} U {2} U {3} U {-} = {0,1,2,3} |  1. Eliminar los estados no alcanzables desde el estado inicial. | |
| **Solución computacional:** | |
| 1. Se crea un Mapa que contiene el conjunto de Estados (HashMap) obtenidos del archivo config.txt y se asigna una clave numérica a cada elemento de la colección en orden ascendente. 2. Se crea un Mapa que contiene el alfabeto de transiciones (HashMap) obtenido del archivo config.txt y se asigna una clave numérica a cada elemento de la colección en orden ascendente 3. Se crea un mapa ordenado (TreeMap) que contiene los datos obtenidos de la función de transición obtenida del archivo config.txt, y se asigna una clave numérica a cada elemento de la colección con la siguiente fórmula.   .   1. En cada clave (Key) del mapa ordenado de transiciones se asigna un conjunto (Set) de elementos que corresponden al elemento de los estado(s) siguiente(s). 2. Si en una clave existe más de un elemento, se realiza la unión de los estados implicados mediante la identificación de estados y transiciones mediante las siguientes fórmulas: 3. Si el conjunto de cada elemento del mapa de transiciones no se encuentra en el mapa de estados, entonces se agrega el conjunto en el mapa de estados. 4. En caso de que surja un nuevo conjunto en el mapa de transiciones, realizar el paso 6 y 7 hasta que todos los estados estén definidos. 5. Se re-etiquetan las claves de los estados nuevos con la fórmula: 6. En caso de que un estado nuevo incluya un estado de aceptación, entonces el nuevo también lo será. 7. Se aplica el componente de estados alcanzables descrito anteriormente.     La representación de los componentes utilizados | |
| **Pruebas** | |
| 1. Original | |  |  | | --- | --- | | ESTADOS | a | | →0 | 0,1,2,3 | | 1 | 2 | | 2 | 3 | | \*3 | - |   Cadenas Aceptadas:   * a * aa * aaa * aaaa   Cadenas No Aceptadas:   * Ø * b * ab * ba * bb |
| * 1. Conversión |
| 1. Original   C:\Users\UAEMex\Desktop\12.jpeg | |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →0 | 1 | - | | 1 | - | 0,2 | | 2 | 1 | - |   Cadenas Aceptadas:   * a * aba * ababa   Cadenas No Aceptadas:   * Ø * abaa * aa * bb * abb |
| 2.1) Conversión |
| 1. OriginalC:\Users\UAEMex\Desktop\12.jpeg | |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →\*0 | 0,1 | - | | \*1 | - | 1,0 |   Aceptadas:   * a * aa * ab * abb * aaa   Cadenas No Aceptadas:   * b * ba * bb * bba * bab |
| 3.1) Conversión  C:\Users\UAEMex\Desktop\Diagram1.jpeg |
| 1. Original   C:\Users\UAEMex\Desktop\1.jpeg | |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →0 | 3,2,4,1 | * - | | 1 | - | 3 | | 2 | - | 4 | | 3 | - | 2 | | \*4 | - | 1 |   Cadenas Aceptadas:   * a * ab * abb * abbb * abbbb * abbbbb   Cadenas No Aceptadas:   * Ø * b * bb * ba * bab |
| 4.1) Conversión  C:\Users\UAEMex\Desktop\Diagram1.jpeg |
| 1. Original   C:\Users\UAEMex\Desktop\2.jpeg | |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →0 | 0,1 | 1 | | \*1 | - | 2,3 | | 2 | 3 | - | | 3 | 0 | - | |
| 5.1) Conversión  C:\Users\UAEMex\Desktop\Diagram1.jpeg | Cadenas Aceptadas:   * a * aa * abb * aba * abab * abaa   Cadenas No Aceptadas:   * ba * bb * bbb * bba |
| 1. Original   C:\Users\UAEMex\Desktop\Diagram1.jpeg | |  |  | | --- | --- | | ESTADOS | a | | →0 | 1,1 | | \*1 | - | |
| 6.1) Conversión  C:\Users\UAEMex\Desktop\Diagram1.jpeg | Cadenas Aceptadas   * a   Cadenas No Aceptadas:   * Ø |
| 1. C:\Users\UAEMex\Pictures\Diagram16.jpegOriginal  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | EDO | a | b | c | d | f | j | V | | →0 | - | - | - | 6 | - | - | - | | \*1 | - | 4,8,10 | - | - | - | - | - | | 2 | 1 | - | 9 | - | - | - | - | | 3 | - | - | - | - | - | - | - | | 4 |  |  |  |  | 5 |  |  | | 5 |  |  |  |  |  | 4 |  | | 6 |  |  |  | 6,10 |  |  |  | | 7 |  |  |  |  |  |  | 10 | | 8 |  |  | 9 |  |  |  |  | | 9 |  |  |  |  |  | 8 |  | | 10 |  |  | 2,3 |  |  |  |  | | 11 |  |  | 8 |  |  |  |  | | |
| 7.1) ConversiónC:\Users\UAEMex\Pictures\62.jpeg |  |
| **Minimización de un Autómata Finito Determinista (AFDM)** | |
| **ENTRADA:** AFD  **SALIDA:** AFDM | |
| 1. El nivel 1 del árbol corresponde al nodo raíz, el cual se forma a partir de todo el conjunto de estados. 2. Para el nivel 2, se creará dos conjuntos a partir del nodo raíz en un conjunto estarán los estados de aceptación y en otro los de no aceptación. 3. Partiendo del nivel se creará el nivel siguiendo el proceso:   Para cada conjunto del nivel hacer:  Para cada símbolo del alfabeto hacer:  Dividir en subconjuntos tales que sus estados , con el símbolo de entrada , tiene una transición a un estado del mismo conjunto .  Para cada símbolo se formó una división particular de conjuntos. Utilizando la división del símbolo anterior, hacer la intersección entre todos los conjuntos para formar la división definitiva.   1. Hacer el paso 3 hasta que no haya subdivisiones en los conjuntos, de lo contrario continuar. 2. Al llegar a este paso, los subconjuntos formados de los pasos anteriores ya no pueden subdividirse más. Si se formó un subconjunto con más de un estado indica que no pueden distinguirse y por lo tanto todos ellos son el mismo estado. Por lo que todos los estados del mismo conjunto son substituidos por el que haya sido escogido como representante de todos ellos. Si el estado inicial se encuentra involucrado en una reducción sería recomendable escogerlo como representante para mantener el mismo estado y evitar confusiones. | |
| **Ejemplo** | |
| C:\Users\UAEMex\Desktop\Nestor\Diagram1.jpeg | |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →1 | 2 | 3 | | 2 | 2 | 4 | | 3 | 2 | 3 | | 4 | 2 | 5 | | \*5 | 2 | 1 | |
| 1. El nivel 1 del árbol corresponde al nodo raíz, el cual se forma a partir de todo el conjunto de estados. 2. Para el nivel 2, se creará dos conjuntos a partir del nodo raíz en un conjunto estarán los estados de aceptación y en otro los de no aceptación. 3. Partiendo del nivel se creará el nivel siguiendo el proceso:   Para cada conjunto del nivel hacer  Para cada símbolo del alfabeto hacer  Dividir en subconjuntos tales que sus estados , con el símbolo de entrada , tiene una transición a un estado del mismo conjunto .  Para cada símbolo se formó una división particular de conjuntos. Utilizando la división del símbolo anterior, hacer la intersección entre todos los conjuntos para formar la división definitiva.   1. Hacer el paso 3 hasta que no haya subdivisiones en los conjuntos, de lo contrario continuar.  |  |  |  | | --- | --- | --- | | ESTADOS | a | b | | →1 | 2 | 3 | | 2 | 2 | 4 | | 3 | 2 | 3 | | 4 | 2 | 5 | | \*5 | 2 | 1 |  1. Al llegar a este paso, los subconjuntos formados de los pasos anteriores ya no pueden subdividirse más. Si se formó un subconjunto con más de un estado indica que no pueden distinguirse y por lo tanto todos ellos son el mismo estado. Por lo que todos los estados del mismo conjunto son substituidos por el que haya sido escogido como representante de todos ellos. Si el estado inicial se encuentra involucrado en una reducción sería recomendable escogerlo como representante para mantener el mismo estado y evitar confusiones. | |
| **Solución Computacional** | |
| 1. Se leen las entradas desde los archivos de texto y se generan   Mapa de Estados  Mapa de Alfabeto  Mapa de transiciones   1. Las transiciones correspondientes a los estados de cada nodo son almacenadas en un *arraylist* de transiciones por estado 2. A partir del mapa de estados se genera una división de estados (de aceptación y no aceptación) y se guardan en la clase *ArrayList* para generar dos nuevos nodos. 3. Se hace una nueva división en el mapa cuando los elementos de un nodo, transitan hacia elementos de otro nodo. 4. Se repite el paso 3 hasta que no queden elementos que transiten hacia otro nodo y mientras los nodos tengan más de un elemento. | |
| **Pruebas** | |
| 1. Original | |  |  |  | | --- | --- | --- | | Estados | a | b | | →0 | 1 | 2 | | 1 | 3 | 4 | | 2 | 0 | 1 | | \*3 | 2 | 5 | | \*4 | 6 | 7 | | 5 | 7 | 1 | | \*6 | 8 | 8 | | 7 | 9 | 5 | | 8 | 7 | 9 | | 9 | 6 | 4 | |
| * 1. Minimización |
| 1. Original | |  |  |  | | --- | --- | --- | | Estados | a | b | | → q0 | q1 | q0 | | \* q1 | q 2 | q1 | | q2 | q2 | q1 | |
| 2.1) Minimización |
| 1. Original | |  |  |  | | --- | --- | --- | | Estados | a | b | | → q0 | q1 | q3 | | \* q1 | q2 | q1 | | q2 | q1 | q2 | | q3 | q4 | q3 | | q4 | q3 | q4 | |
| 3.1) Minimización |
| **Validación de cadenas** | |
| **ENTRADA:** Autómata finito determinista con cadenas separadas por comas  **SALIDA:** Cadenas válidas con una etiqueta “True” y las no válidas con una etiqueta “False”   1. Dada una colección de cadenas, estás son validadas de acuerdo a un autómata establecido. 2. Se obtiene la validación conforme a la transición de cada uno de los elementos del alfabeto que existe dentro del autómata descrito. 3. Si la cadena transita hasta un estado de aceptación se puede deducir que la cadena es válida, de lo contrario, tal vez la cadena no ha terminado de transitar o quedo en un estado que no es de aceptación, entonces se deduce que la cadena no es válida. | |
| **Ejemplo** | |
| 1. Dada una colección de cadenas, estás son validadas de acuerdo a un autómata establecido.   Juan Carlos Es Alto   1. Se obtiene la validación conforme a la transición de cada uno de los elementos del alfabeto que existe dentro del autómata descrito.   C:\Users\UAEMex\Pictures\Diagram1.jpeg   1. Si la cadena transita hasta un estado de aceptación se puede deducir que la cadena es válida, de lo contrario, tal vez la cadena no ha terminado de transitar o quedo en un estado que no es de aceptación, entonces se deduce que la cadena no es válida.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Estados | Juan | Carlos | Es | Alto | | →0 | 1 | - | - | - | | 1 | - | 2 | - | - | | 2 | - | - | 3 | - | | 3 | - | - | - | 4 | | \*4 | - | - | - | - |     Juan Carlos Es Alto = “True” | |
| **Solución Computacional** | |
|  | |
| **Pruebas** | |
|  | |
|  |  |