Cálculo del Autómata Universal

Para realizar la práctica 2 de Teoría de Lenguajes, se ha seguido usando la clase AFD implementada para la práctica anterior, y hemos añadido una serie de métodos nuevos para el cálculo del Autómata Universal de un AFD dado.

Además de esta clase, será necesario implementar una clase de Autómatas Finitos No Deterministas (AFN), ya que el autómata universal calculado es un autómata de esta clase. La implementación es muy similar a el AFD, eliminando los métodos implementados en la primera práctica, y cambiando la estructura de datos que almacena las transiciones, para que soporte transiciones con el mismo símbolo a diferentes estados.

Se ha implementado el algoritmo de cálculo de Autómatas Universales visto en clase, que realiza en primer lugar en el calculo del reverso del autómata, y el determinista de este.

Tras obtener este autómata, te toman los estados, formados por conjunto de estados del AFD original, y junto a todas las posibles intersecciones de los estados originales, se construye una tabla, donde los elementos de esta son 1's o 0's según el estado este presente en el conjunto de estados del determinista del reverso, o no. Para las intersecciones será 1 ó 0 según el resultado de la operación AND sobre los estados implicados.

De la tabla se extraen los estados que sean distinguibles, y se establece las relaciones de inclusión entre los estados. Para construir el autómata universal se parte de las transiciones iniciales y se añaden las transiciones de la siguiente forma: si llega una transición a un estado, esta también llega a los estados que están incluidos en este.

Método *calculaEstadosDR*

En el algoritmo inicial, es especifica que se tiene que realizar el reverso del autómata, y luego hallar su versión determinista, pero no es necesario obtener todo el autómata, sino únicamente los conjuntos de estados.

En primer lugar se calcula las transiciones que corresponderían al autómata reverso, así como los estados iniciales, correspondientes a los finales del AFD original.

Nos guardamos en un un conjunto de conjuntos de estados, los estados iniciales del reverso, es decir, los finales del original, que será el primer conjunto de estados que tendremos que procesar.

Mientras no nos queden estados por procesar, cogemos el primer estado que nos quede por procesar, y sólo si este no esta ya incluido en la lista de estados finales.

Para cada símbolo, y dentro de este para cada sub-estado del estado actual, nos guardamos las transiciones que van de este sub-estado con el símbolo en el autómata original, y nos guardamos el destino de estas. Nos guardamos la transición, y si el destino no es vacio, y es distinto del estado actual y no esta en el conjunto de estados resultantes, nos guardamos el destino en la lista de estados por procesar.

Por último nos guardamos el estado actual en el conjunto de estados resultantes, y seguimos iterando, mientras no queden estados pendientes.

Método *AutomataUniversal*

En este método se parte del resultado del método *calculaEdadosDR*, del cual se obtienen los estados del determinista del reverso.

Una vez tenemos estos estados, y con los estados del autómata original, comenzamos rellenando la primera parte de la tabla, poniendo en un vector de *booleanos*, de tamaño el numero de estados del determinista del reverso, poniendo a true si el estado pertenece al conjunto, y false en caso contrario.

Después se generan las intersecciones entre estados. Para esto, usamos una estructura **set** de **set's de enteros**, donde los elementos no pueden estar repetidos, y generamos las intersecciones de forma incremental. Por ejemplo, si hay 3 estados, {1, 2, 3} se generarán las intersecciones de la siguiente forma:

1∩1, 1∩2, 1∩3, 2∩1, 2∩2, 2∩3, 3∩1, 3∩2, 3∩3, 1∩1∩1, 1∩1∩2, 1∩1∩3, 1∩2∩1, ...

Pero como no se permiten conjuntos repetidos, solo se almacenarán las siguientes intersecciones:

1∩2, 1∩3, 2∩3 y 1∩2∩3

Esta parte del algoritmo tiene un coste exponencial de $O(n^n)$.

Una vez tenemos las intersecciones válidas, calculamos los vectores de *booleanos* asociados a a cada uno de estas, aplicando el operados AND entre los vectores correspondientes a los estados implicados en la intersección.

Tanto en este punto, como en de la generación de los vectores de los estados originales, se mantiene una estructura de datos para almacenarlos que evita repeticiones, para así ya tener en este punto todos los vectores que sean distinguibles.

El último paso antes de generar el autómata propiamente dicho, es establecer las relaciones de inclusión entre los estados. Para esto analizamos los vectores calculados para los estados y las intersecciones, y se guarda para cada estado, los que están incluidos.

Las transiciones finales vienen dadas por las transiciones del autómata original, a las que añadimos en primer lugar aquellas que salen de los nuevos estados. Estas son aquellas que partiendo de cada uno de los estados implicados en la intersección, con el mismo símbolo llegan al mismo estado. En segundo lugar añadimos transiciones de tal forma que si una transición llega a un estado, esta llega también a todos lo que lo contienen.

Ya para terminar construimos el nuevo autómata con los estados y transiciones finales que se han calculado, y con el mismo alfabeto y estados finales e iniciales del autómata original.

Algunos ejemplos de ejecución

Para todos los ejemplos: Σ = {0, 1}

Entrada:

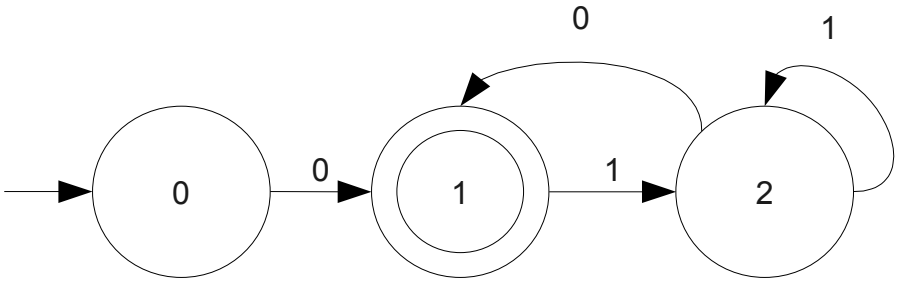
Q = {0, 1, 2} F = {1} q_0 = 0

$\delta(0,0) = 1$

$\delta(1,1) = 2$

$\delta(2,0) = 1$

$\delta(2,1) = 2$



Salida:

Q = {0, 1, 2, 3} F = {1} q_0 = 0

$\delta(0, 0) = 1$

$\delta(0, 0) = 3$

$\delta(1, 1) = 0$

$\delta(1, 1) = 2$

$\delta(1, 1) = 3$

$\delta(2, 0) = 1$

$\delta(2, 0) = 3$

$\delta(2, 1) = 0$

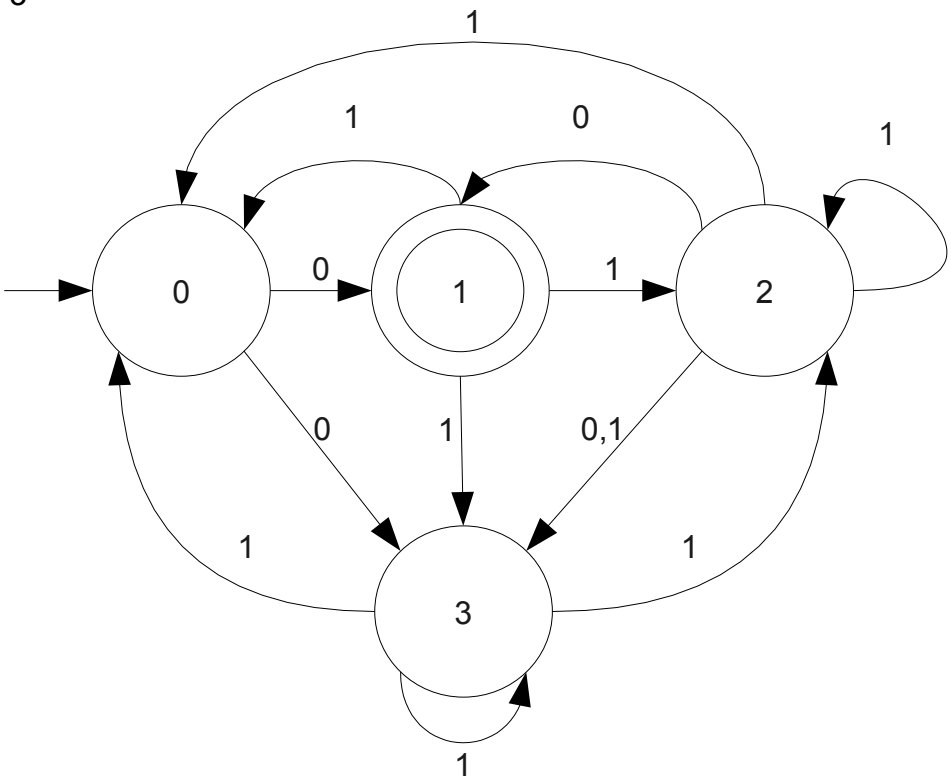
$\delta(2, 1) = 2$

$\delta(2, 1) = 3$

$\delta(3, 1) = 0$

$\delta(3, 1) = 2$

$\delta(3, 1) = 3$



El estado 3 es el equivalente para el estado $1 \cap 2$, que en la implementación se renombra los estados intersección para hacerlos compatibles con el resto de métodos.

Medidas de complejidad de ejecución

El algoritmo implementado para el cálculo del Autómata Universal tiene un coste exponencial de $O(n^n)$ a causa del proceso de computo de todas las posibles intersecciones entre los estados.

Este coste se puede apreciar también de forma empírica en la siguiente gráfica realizada midiendo el tiempo de ejecución del algoritmo para autómatas de *n* estados, siendo n = { 3, 4, 5, 6, 7}.