AUTLEN – Memoria Práctica 3

Grupo 10
Tomás Higuera Viso & Andrés Calvente Rodríguez

En esta práctica hemos tenido que expandir el modulo que anteriormente habíamos estado trabajando para que pudiésemos crear autómatas finitos no deterministas a partir de una expresión regular. El programa no lee una cadena en la cual tengamos una expresión regular, si no que el usuario tiene las herramientas necesarias para crear paso a paso la expresión regular que el haya pensado. Las funciones que hemos codificado son las siguientes:

- ***AFND1ODeSimbol(char* simbolo)***

Esta función crea una AFND con dos estados inicial y final los cuales están unidos por el símbolo especificado por el argumento de entrada. La función utiliza las funciones propias del modulo para crear AFND, insertar estado, insertar transición, etc.

- ***AFND1ODeLambda()***

Esta función crea una AFND con dos estados inicial y final los cuales están unidos por una transición de tipo Lambda.

- ***AFND1ODeVacio()***

Esta función crea una AFND con un único estado de tipo inicial y final.

- ***AFNDAAFND1O(AFND* p_afnd)***

Esta función transforma un AFND normal, con varios estados iniciales y finales, a un AFND con un solo estado inicial y un solo estado final. Para ello, dentro de la función se crea un AFND nuevo, inicializando todos los estados y todas las transiciones a las propias del AFND pasado por argumento y después se crean dos estados nuevos los cuales van a ser los estados inicio y final. Además, tenemos que procurar que el nuevo estado inicial apunte con transiciones Lambda a los antiguos iniciales y los antiguos finales apunten de la misma forma al nuevo estado final.

- ***AFND1OUne(AFND* p_afnd1O_1, AFND* p_afnd1O_2)***

Esta función crea un AFND juntando los dos AFNDs pasados como argumento, esto hace que se puedan reconocer las palabras que genera el AFND1 como las que genera el AFND2. Para crear el nuevo AFND tenemos que tener en cuenta que los estados totales son la suma de los dos originales, y el número de símbolos serán los que contiene el AFND1 más los que tenga el AFND2 que no tenga el primero. Deberemos de insertar todos los estados y asegurarnos de nuevo de no meter símbolos repetidos en el alfabeto. También tendremos que insertarlas transiciones asegurándonos de que estamos cogiendo bien los estados inicial y final de la transición y los símbolos con los que transita. Finalmente llamaremos a la función anteriormente comentada ***AFNDAAFND1O( )*** para que convierta nuestro AFND nuevo en uno con una sola entrada y una sola salida, terminando así la unión entre AFNDs.

- ***AFND1OConcatena(AFND* p_afnd_origen1, AFND* p_afnd_origen2)***

Esta función concatena dos AFNDs poniendo entre medias de los dos una transición Lambda. El procedimiento de codificación de la función es el mismo que el de la función anteriormente comentada.

- ***AFND1OEstrella(AFND* p_afnd_origen)***

Función que pasa un AFND normal a un con propiedad estrella, la cual hace que el autómata reconozca la palabra vacía. Para ello se realizará el mismo procedimiento de codificación explicado en las funciones anteriores con la salvedad que al finalizar con la llamada de la función ***AFNDAAFND1O( )*** tendremos que añadir 2 nuevas transiciones Lambda, una que vaya del estado inicial al final y otra que vaya del final al inicial.

- ***AFNDADot(AFND* p_afnd)***

Función que crea el archivo *.dot* propio al AFND pasado como argumento.