Práctica 3

Autómatas y lenguajes

AFND equivalentes a ER

Autores:

Francisco de Vicente Lana
francisco.vicentel@estudiante.uam.es

Ricardo Riol González
ricardo.riol@estudiante.uam.es

Grupo 1401

13 de diciembre de 2018

Profesor: Mariano Rico

Índice

1. Diseño

En esta práctica, añadimos a nuestra librería de autómatas la funcionalidad necesaria para generar nuevos autómatas a partir de otros más pequeños.Comentar que hemos seguido los algoritmos de la transparencias, salvo en el caso de la operación de concatenar.

En el caso de concatenar, no añadimos dos estados nuevos al principio y al final , si no que , suponemos que el inicial del autómata resultado es el estado inicial del primer autómata a concatenar y el final del autómata resultado es el final del segundo autómata a concatenar. Optamos por esta opción, porque se hace un uso más prudente de la memoria, consiguiendo los mismos resultados.

Para representación de los autómatas, hemos utilizado la herramienta DOT. Es de gran importancia, comentar que realizamos los dibujos de estos antes de realizar el cierre de transiciones lambda, ya que que si se realizase después, la representación quedaría muy liosa.

2. Modo de ejecución

Para ejecutar el programa, es conveniente seguir las siguientes indicaciones:

- Hacer **make**
- Ejecutar cualquiera de los test proporcionados, que generará automáticamente las imágenes de los autómatas. El test parseará varias cadenas de la expresión regular para ver si las acepta.
- En el interior del directorio de la entrega, existe un directorio imágenes donde se guardan las imágenes de los autómatas cada vez que se genera un test.

3. Pruebas

3.1. Test 1

Se ha pasado correctamente el índice proporcionado en el enunciado. A continuación, se adjunta una imagen de los autómatas que se generan de este test.

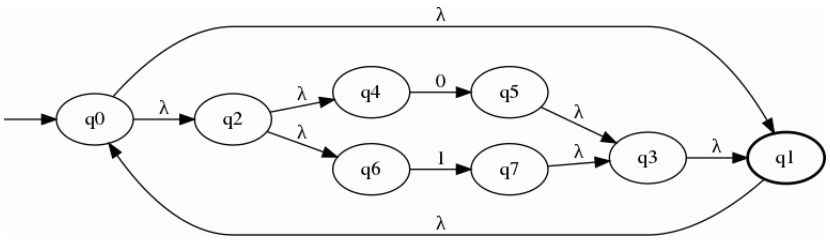

Figura 1: Test1((0+1)*)

3.2. Nuestras pruebas

3.2.1. Test 2

En este test probamos, si nuestra codificación permite realizar varias veces la operación * sobre un autómata. Debería tener el mismo resultado que si se realizase solo una vez.

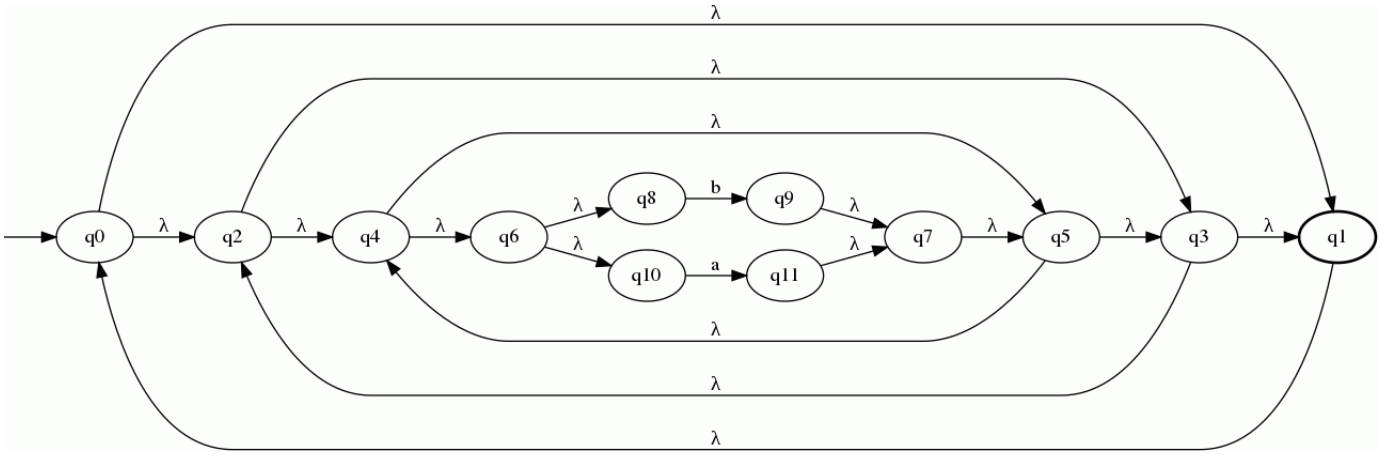

Figura 2: Test1((0+1)***)

3.2.2. Test 3

En este test probamos a realzar varias concatenaciones y sumas en un solo autómata para ver si se realizan correctamente. Se prueba el autómata correspondiente a la siguiente expresión regular: (00)+(01)+(10)+(11).

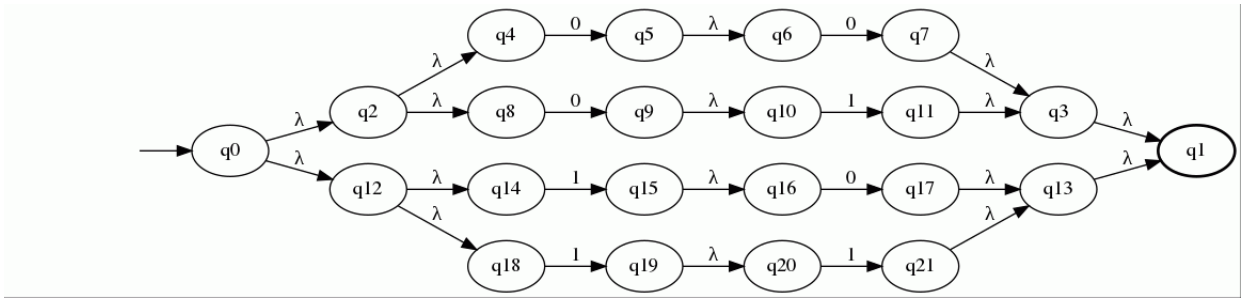

Figura 3: Test3

3.2.3. Test 4

En este test, probamos la función AFNDAAFND1O que coge cualquier autómata con varios estados finales y los resume en uno solo. Para ello, cogemos un autómata con varios iniciales y finales:

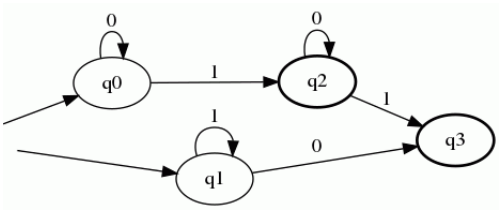

Figura 4: Test4

y después de aplicar la función AFNDAFND1O obtenemos el siguiente autómata:

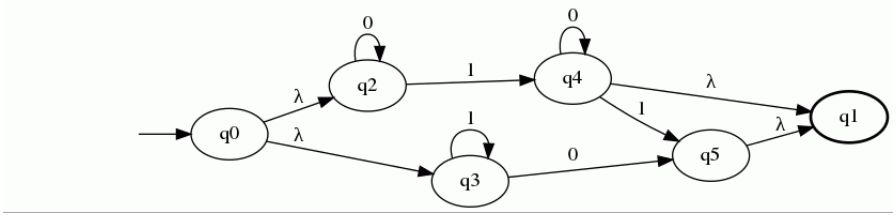

Figura 5: Test4

4. Conclusión

Por último, cabe destacar que de nuevo hemos hecho uso de herramientas auxiliares como Github, valgrind (memcheck), gdb o latex.