Práctica 2

Autómatas y lenguajes

Simulación de autómatas finitos no deterministas

Autores:

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

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

Grupo 1401

12 de noviembre de 2018

Profesor: Mariano Rico

Índice

1. Diseño

En esta práctica hemos añadido la funcionalidad necesaria para tratar transiciones lambda. Primero explicamos la función que permite cerrar la transitividad inducida entre varios estados por transiciones lambda.
transicion_inducir_aux es la encargada de inducir todos los estados que son accesibles mediante transiciones lambda desde un estado que se le pasa como argumento. Esta función trabaja con dos índices: **i** (estado a investigar) y **pos** (primera posición vacía). Si **i** y **pos** son iguales significa que no hay más estados que investigar. Mientras sean distintos, miramos las conexiones lambda inmediatas del estado que ocupa la posición i-ésima del array y los añadimos a la lista aumentando el valor de pos y de i. De esta forma conseguimos, a partir de un estado inicial, todos los estados accesibles por transiciones lambda.
transicion_inducir por su parte es la encargada de, por cada uno de los estados del autómata, llamar a la función anterior, y escribe los nuevos estados accesibles en la tabla de transiciones lambda.
En cuanto a la función de **procesar** una cadena, modificamos que cada vez que se incluye un estado a la lista para procesar, también se añaden todos aquellos accesibles por transiciones lambda.

Ejemplo de funcionamiento (autómata del enunciado)

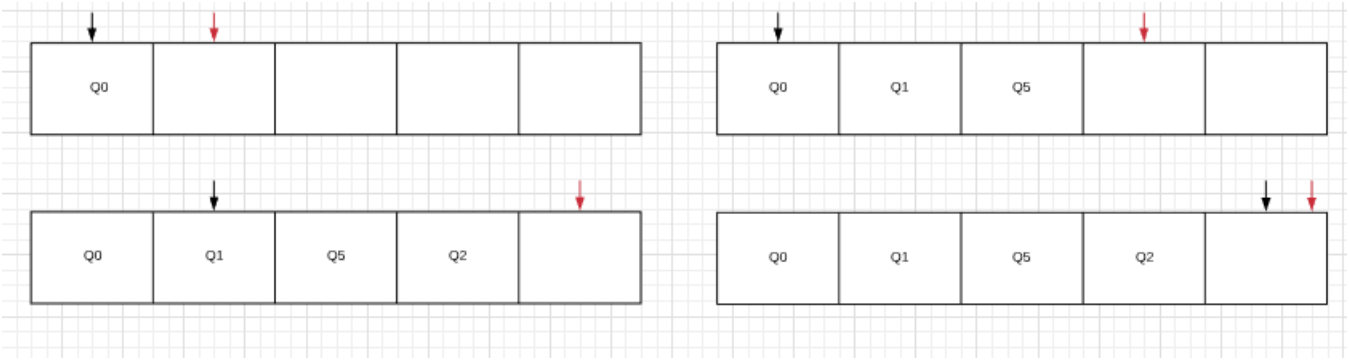

Figura 1: Ejemplo de funcionamiento

2. Pruebas y comentarios

En esta ocasión hemos realizado 5 programas de pruebas los cuales simulan autómatas con distintas propiedades. A continuación los comentamos brevemente:

- **test1**: se trata del proporcionado en el enunciado y funciona como cabría esperar.
- **test2**: es un ejemplo de un autómata sencillo con una única transición lambda visto en clase de teoría.
- **test3**: este caso incorpora un bucle de transiciones lambda de forma que nos aseguramos que nuestro algoritmo lo trata adecuadamente.
- **test4**: cubre la posibilidad de que haya un estado final e inicial al mismo tiempo.
- **test5**: es el más completo, ya que incluye un alfabeto con tres caracteres y cuenta con tres estados enlazados sucesivamente por transiciones lambda, con lo cual probamos la circunstancia de que haya transitividad entre más de dos estados. Por ello analizamos este caso en detalle.

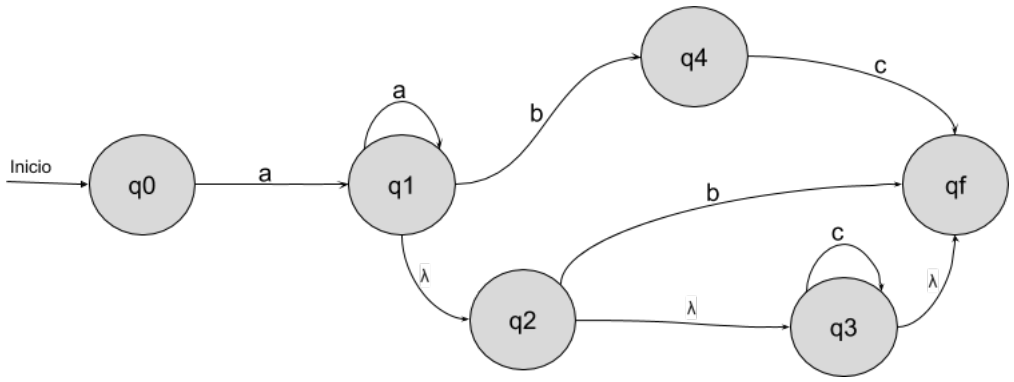

Figura 2: afnd del test4

Probamos las cadenas **a** para ver que es aceptada gracias a que funciona el enlace múltiple lambda q1-qf, **abc** que comprueba la rama superior; y los casos rechazados (vacío), **ba, aaabb y aba**.

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