A.M.C.

Memoria Práctica 2:

Autómatas Finitos

Autores:

Miguel Quiroga Campos
Daniel Linfon Ye Liu

Curso 2023/2024

AFD

Un autómata finito determinista (AFD) es un modelo matemático que representa un sistema con un número finito de estados, transiciones entre ellos y un conjunto finito de símbolos de entrada. En cada paso, el AFD lee un símbolo de entrada y se mueve de un estado a otro según una función de transición predefinida. Su comportamiento es determinista, ya que para un estado y un símbolo de entrada dados, solo hay una transición posible. Si el AFD alcanza un estado final después de procesar una cadena de entrada, se acepta; de lo contrario, se rechaza.

A continuación se muestra el funcionamiento de nuestra implementación de un AFD en Java, para el siguiente fichero de entrada:

```
TIPO: AFD
ESTADOS: q1 q2 q3 q4 q5 q6
INICIAL: q1
FINALES: q4 q5 q6
TRANSICIONES:
q1 'a' q2
q2 'b' q4
q4 'b' q4
q3 'a' q5
q5 'a' q6
q2 'a' q3
q5 'b' q2
FIN
```

Para empezar, tendremos la opción de elegir cargar un fichero existente o de crear uno nuevo. Para mostrar un funcionamiento más detallado de nuestra solución, escogeremos la opción de introducir el autómata manualmente, esto creará un fichero idéntico al anterior y cargará automáticamente el AFD en la memoria de la aplicación para poder usarlo.

El programa iniciará solicitando al usuario el número de estados, con indicación del estado inicial y los estados finales. Posteriormente, se solicitará la introducción de los estados restantes hasta alcanzar la cantidad previamente especificada.

```
Introduce el número de estados:
6
Introduce el estado inicial:
q1
Introduce el número de estados finales:
3
Introduce el estado final 1:
q3
Introduce el estado final 2:
q4
Introduce el estado final 3:
q5

***** Faltan 2 estados por definir *****

Introduce el estado 1:
q2
Introduce el estado 2:
q6
```

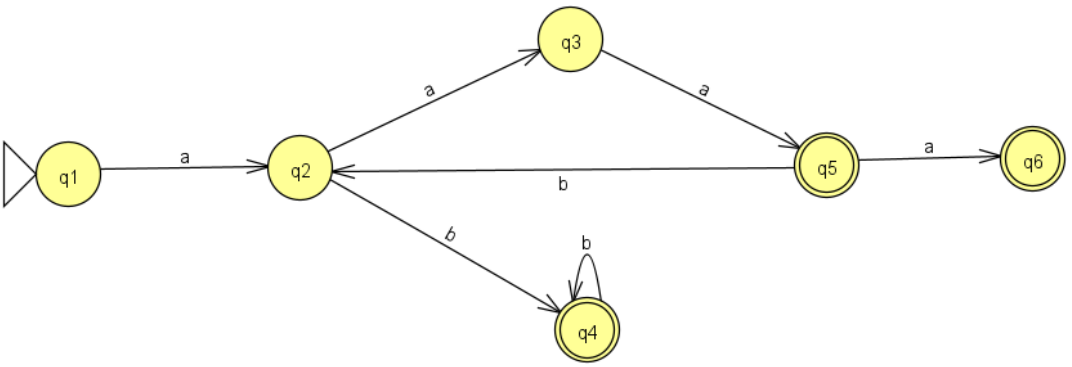
A continuación, se consultará si se desea agregar transiciones. En caso afirmativo, se requerirá al usuario que especifique el estado de origen, el estado de destino y el símbolo correspondiente para cada transición. Esta operación podrá repetirse según la cantidad de transiciones que se deseen agregar.

```
¿Quiere añadir una transición? (s/n)
s

***** TRANSICIÓN 1 *****

        - Estado origen:
q1
        - Símbolo de transición:
a
        - Estado destino:
q2

¿Quiere añadir otra transición? (s/n)
s

***** TRANSICIÓN 2 *****

        - Estado origen:
q2
        - Símbolo de transición:
b
        - Estado destino:
q4

¿Quiere añadir otra transición? (s/n)
```

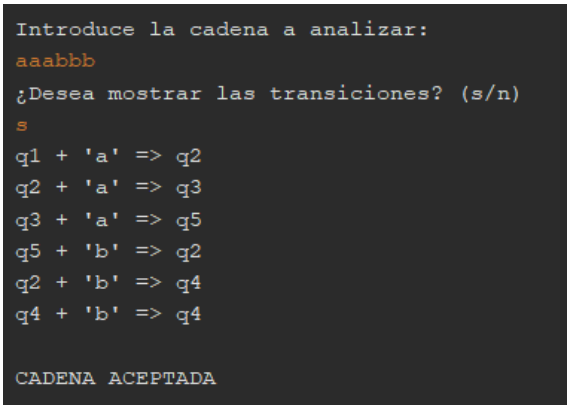
Finalmente, una vez ingresadas todas las transiciones, el programa generará un archivo con el formato previamente mencionado. Este archivo será cargado en la memoria de la aplicación, listo para su manipulación.

Mencionar que durante todo el proceso de recogida de datos, la aplicación poseerá la capacidad de detectar posibles errores en la información introducida, solicitando la corrección de los datos erróneos.

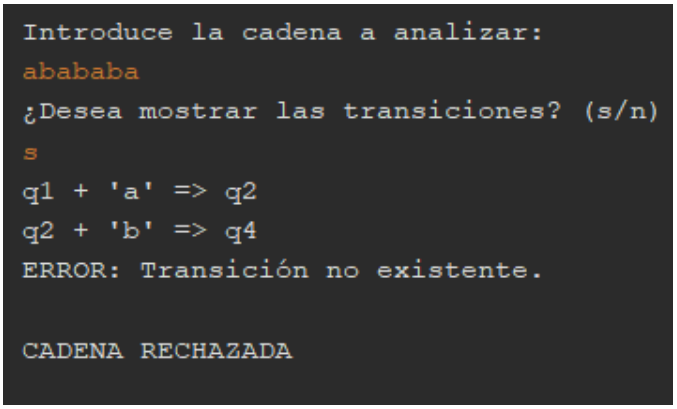
Nuestro fichero se podría representar gráficamente de esta manera:



La aplicación nos pedirá introducir una cadena a analizar y nos indicará si la acepta o la rechaza. Además, nos ofrecerá la posibilidad de mostrar o no las transiciones realizadas durante el recorrido.



Ejemplo de Cadena CORRECTA



Ejemplo de Cadena INCORRECTA

AFND

Un autómata finito no determinista (AFND) es un modelo matemático que, a diferencia del AFD, puede tener múltiples transiciones posibles para un estado y un símbolo de entrada dados. Esto significa que en un estado dado, el AFND puede no determinar una única transición para un símbolo específico, lo que lo hace más flexible en su comportamiento. Durante el procesamiento de una cadena de entrada, el AFND puede seguir varios caminos simultáneamente, explorando diferentes opciones. Si al menos uno de estos caminos conduce a un estado final al finalizar la cadena, el AFND acepta la entrada.

A continuación se muestra el funcionamiento de nuestra implementación de un AFND en Java, para el siguiente fichero de entrada:

```
TIPO: AFND
ESTADOS: q1 q2 q3 q4
INICIAL: q1
FINALES: q3 q4
TRANSICIONES:
q1 'a' q2 q3
q2 'b' q3
q2 'a' q2
q3 'a' q1 q4
q4 'b' q2 q4
TRANSICIONES LAMBDA:
q1 q1
q4 q4 q3
FIN
```

De nuevo, tenemos la posibilidad de elegir si cargar un fichero existente o de crear uno nuevo. Volveremos a escoger la opción de crear uno nuevo, que tiene un funcionamiento similar al de la AFD.

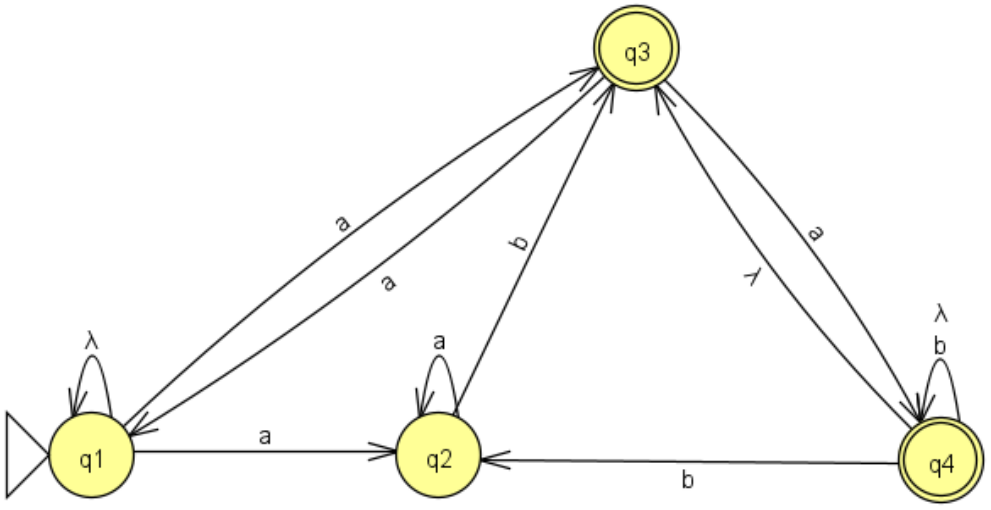
```
¿Quiere añadir una transición? (s/n)
s

***** TRANSICIÓN 1 *****

        - Estado origen:
q1
        - Símbolo de transición:
a
Introduce el número de estados destino:
2
        - Estado destino 1:
q2
        - Estado destino 2:
q3

¿Quiere añadir otra transición? (s/n)
```

En este caso, al ser un AFND, a la hora de añadir las transiciones podemos indicar el número de estados destinos. Además, al finalizar con las transiciones, nos preguntará si deseamos introducir transiciones lambda.

```
¿Quiere añadir una transición LAMBDA? (s/n)
s

***** TRANSICIÓN LAMBDA 5 *****

        - Estado origen:
q1
Introduce el número de estados destino:
1
        - Estado destino 1:
q1
```

Una vez introducidos todos los datos, se habrá generado un fichero como el que hemos mostrado previamente y estará cargado en la memoria de la aplicación.

El grafo que representa nuestro fichero sería este:



En este caso, adicionalmente, la aplicación nos mostrará todos los estados actuales en los que se encuentra el autómata, con idea de facilitar el seguimiento de la ejecución.

```
Introduce la cadena a analizar:
aabbaa
¿Desea mostrar las transiciones? (s/n)
s
Estados actuales: q1

q1 + 'a' => q2 q3
Estados actuales: q2 q3

q2 + 'a' => q2
q3 + 'a' => q1 q4
Estados actuales: q1 q2 q3 q4

q2 + 'b' => q3
q4 + 'b' => q2 q4
Estados actuales: q2 q3 q4

q2 + 'b' => q3
q4 + 'b' => q2 q4
Estados actuales: q2 q3 q4

q2 + 'a' => q2
q3 + 'a' => q1 q4
Estados actuales: q1 q2 q3 q4

q1 + 'a' => q2 q3
q2 + 'a' => q2
q3 + 'a' => q1 q4
Estados actuales: q1 q2 q3 q4
CADENA ACEPTADA
```

Ejemplo de Cadena CORRECTA

```
Introduce la cadena a analizar:
abbbaaa
¿Desea mostrar las transiciones? (s/n)
s
Estados actuales: q1

q1 + 'a' => q2 q3
Estados actuales: q2 q3

q2 + 'b' => q3
Estados actuales: q3

Estados actuales:

Estados actuales:

Estados actuales:

Estados actuales:

Estados actuales:
CADENA RECHAZADA
```

Ejemplo de Cadena INCORRECTA

Conclusiones

En resumen, mientras que un AFD tiene reglas de transición bien definidas y únicas para cada combinación de estado y símbolo de entrada, un AFND permite cierta flexibilidad al tener múltiples opciones de transición o la posibilidad de no realizar ninguna transición en un estado dado.

Ambos tipos de autómatas tienen sus propias ventajas y desventajas, y son utilizados en diversos contextos según las necesidades del problema que están diseñados para resolver.

En resumen, los AFD son preferibles cuando se busca simplicidad y determinismo, y cuando el comportamiento del sistema es predecible. Los AFND son útiles cuando se necesitan representar comportamientos no deterministas y cuando la flexibilidad en las transiciones es esencial.