PRÁCTICA 2
AMC

Grupo: AMC-L2

Ismael Da Palma Fernández

ÍNDICE

Implementación de la práctica:

En esta práctica se nos pedía desarrollar una "aplicación" de autómatas finitos deterministas (AFD) y autómatas finitos no deterministas (AFND), los datos del autómata tenemos que recogerlos o bien por fichero o bien por teclado.

Para la parte de programación de la práctica, he creado las siguientes clases:

- **AFD:** clase la cual se encargará de administrar las diferentes operaciones que realicemos con un autómata AFD.

  Esta clase implementa la interfaz Cloneable (ya incluida en Java) y la interfaz Proceso (creada manualmente) para que contenga algunos métodos comunes entre ambos autómatas (clone(), toString(), reconocer(), esFinal()).

  Los métodos más destacados de esta clase son:
  - **agregarTransición(es1, símbolo, es2):** que se encarga de añadirle una nueva transición a la lista de transiciones del autómata, antes de añadirla se encargará de comprobar que la transición que se quiere insertar no exista en las actuales del autómata.

  - **transición(es, symbol):** este método devolverá el estado al que pasa el autómata cuando se encuentra en el estado "es" y transitando con el símbolo "symbol" .

  - **esFinal(es):** comprueba que el estado "es" pertenece al array de String de estados finales. Si pertenece devolverá true, en caso contrario devolverá false.

  - **reconocer(cadena):** el método más importante de la clase, ya que se encargará de comprobar que la cadena que recibe por parametro es aceptada o rechazada por el autómata que hayamos generado, este método de encargará de ir llamando constantemente al método "transición()" hasta que la cadena llegue a su fin y se comprobará con "esFinal()" si el estado en el que se quedó el autómata es final o no, si lo es, la cadena será aceptada.

- **AFND:** esta clase se encargará de administrar las diferentes operaciones que se realicen con un autómata AFND.
  Esta clase también implementa la interfaz Cloneable y Proceso tal y como lo hace la clase AFD.

  Los métodos más destacado de esta clase son:
  - **agregarTransición(es1, simbolo, es2):** este método es idéntico al método de agregar transición del AFD.

  - **agregarTransicion λ(es1, es2):** similar al método anterior, salvo que esta vez, la transición se añade a la lista de transiciones λ del autómata AFND.

  - **transición(macroestado, simbolo):** el método devuelve el conjunto de estados al que puede ir el autómata con el simbolo pasado por parametro partiendo de los estados que nos pasan.

  - **transición λ(estado):** devuelve los estados a los que puede saltar el autómata a partir del que le pasan por parametro sin llegar a gastar un símbolo.

  - **esFinal(estado):** método con la misma funcionalidad que la del AFD.

- **λ_clausura(macroestado):** el método más destacado de la clase se encarga de buscar las transiciones λ de cada uno de los estados que se pasa por parametro y las devuelve junto con estos.

- **reconocer(cadena):** tiene la misma funcionalidad que la del AFD.

- **Controlador:** esta es la clase que se va a encargar de administrar las diferentes interfaces que utiliza la práctica y también de controlar los eventos que activa cada una de estas.

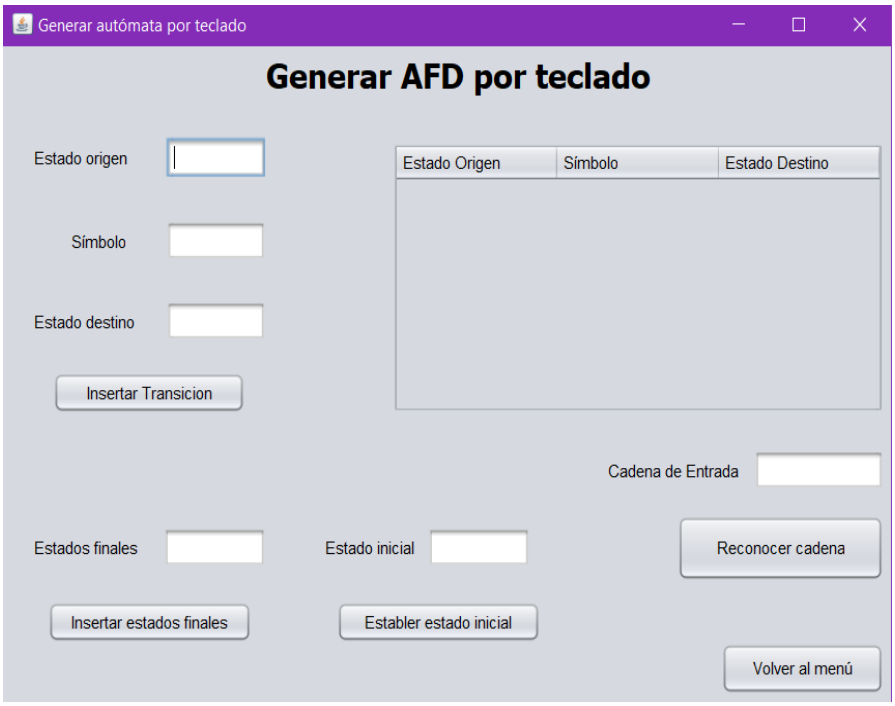
- **Ficheros:** es la clase que se encarga de la parte de los ficheros de la práctica, aunque tenga varios métodos implementados, no se con certeza si estos funcionan ya que en la Moodle no había ningún fichero de autómatas de ejemplo para probar esos métodos.

- **UtilTablas:** clase que utiliza el Controlador para ayudarse a rellenar, vaciar y configurar las tablas de las diferentes interfaces, estas tablas mostraran información sobre las transiciones del autómata que estemos generando por teclado.

Para finalizar la parte de programación voy a generalizar las clases de "TransicionAFD/AFND/ λ" ya que todas ellas tienen una estructura similar, sólo tienen métodos getters y setters y un constructor y son usadas por las clases AFD (solo usa la transicionAFD) y AFND (usa la transicionAFND y transicion λ). Cada una contiene las transiciones que va a usar el autómata correspondiente.

Con respecto a la parte gráfica de la práctica he creado cuatro JFrames diferentes que utilizaré dependiendo de qué forma quiera generar el autómata y que autómata generar.

Vamos a ver sólo las interfaces por teclado:



Para establecer una transición debemos de insertar un estado origen, un símbolo y un estado destino.
Si le damos a **Insertar Transición**, llamará al método "agregarTransicion()" del autómata.

En estados finales sólo se nos permite insertar estados de 1 en 1 (debido a la implementación, esto se corrige en el AFND).
Al darle al botón de **Insertar estados finales**, llamaremos desde el controlador al método "setEstadosFinales()" del autómata (similar ocurre con el botón de establecer estado inicial).

El botón de **reconocer cadena** recoge lo escrito en "cadena entrada" y llamará al método reconocer() del AFD.

Tenemos tres formas de insertar una transición AFND en esta interfaz.

-Si el estado origen solo transita con transicionesλ, se rellenarán solo el campo de "Estado Origen" y "Transición λ".

-Si el estado origen transita con un símbolo a un/unos estados, se rellenarán solo los tres primeros campos.

-Si el estado origen transita con un símbolo y además con transicionesλ, debemos de rellenar todos los campos.

El resto de los botones tiene el mismo uso que el de la interfaz del AFD por teclado, solamente que este llamará a los métodos correspondientes del AFND.

Principales problemas y soluciones a estos:

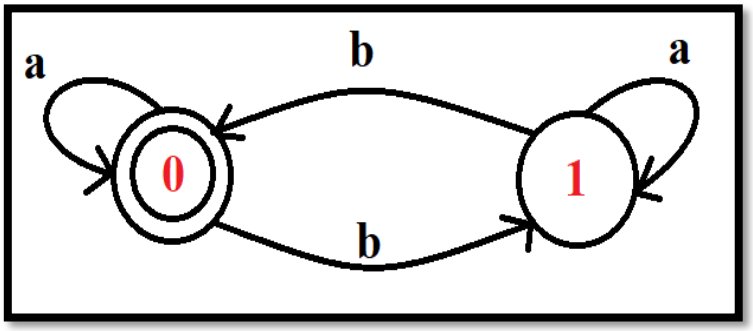
Con respecto a la implementación del AFD no he tenido ningún problema, mientras que la implementación del AFND me ha generado numerosos de ellos como que me fallaba a la hora de mostrar el conjunto de estados destino o transicionesλ en la tabla de la interfaz por teclado del AFND. Al parecer lo que estaba haciendo era mostrar el array de Strings completo en un String (que es lo que guarda la columna de la tabla). Para resolverlo solo tuve que hacer un bucle for demás en el método "rellenarTablaPorTecladoAFND" dentro del otro bucle que recorría las diferentes filas de la tabla.

Otro problema que he tenido ha sido a la hora de implementar el método λ_clausura básicamente porque me resultaba un poco complejo a la hora de empezar a codificarlo, pero con esfuerzo y esmero creo que conseguí implementar sin problemas el método, aunque me hubiese gustado hacerlo en menos líneas.

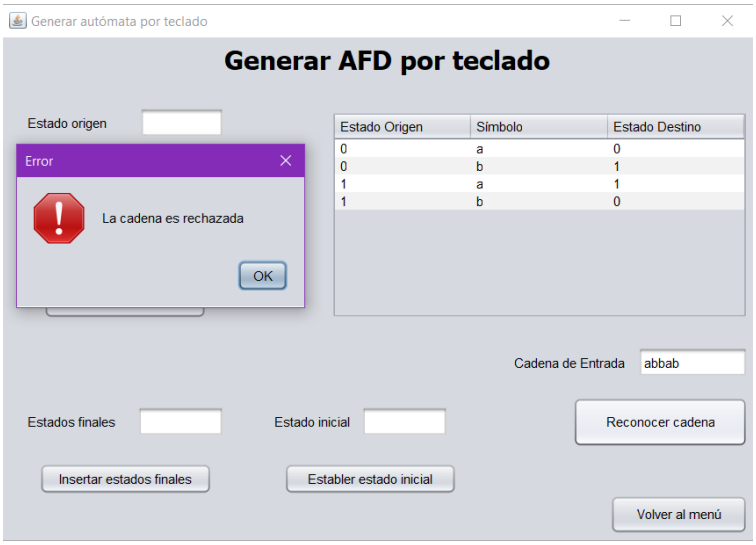
Ejemplos de ejecución de autómatas:

EJEMPLOS AFD

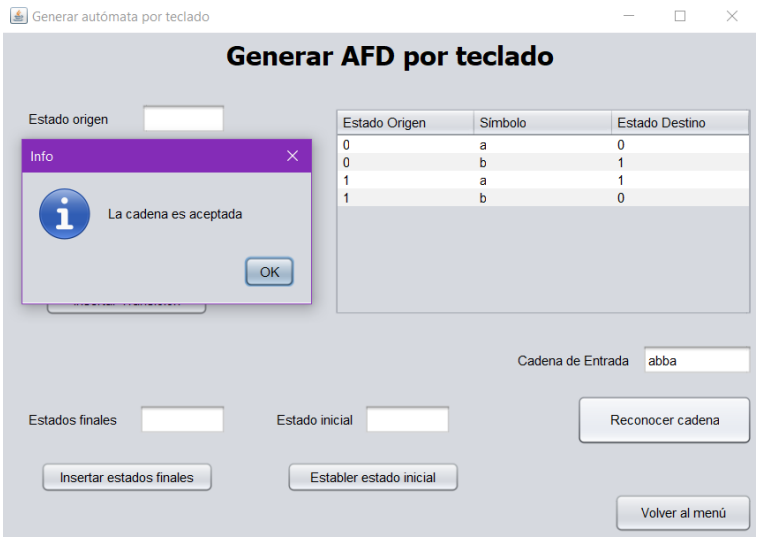
- Ejemplo 1: un grafo simple



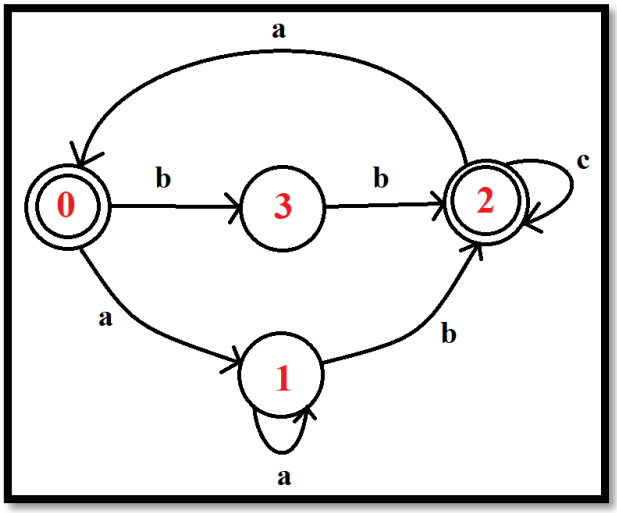
Cadena: abbab -> rechazada



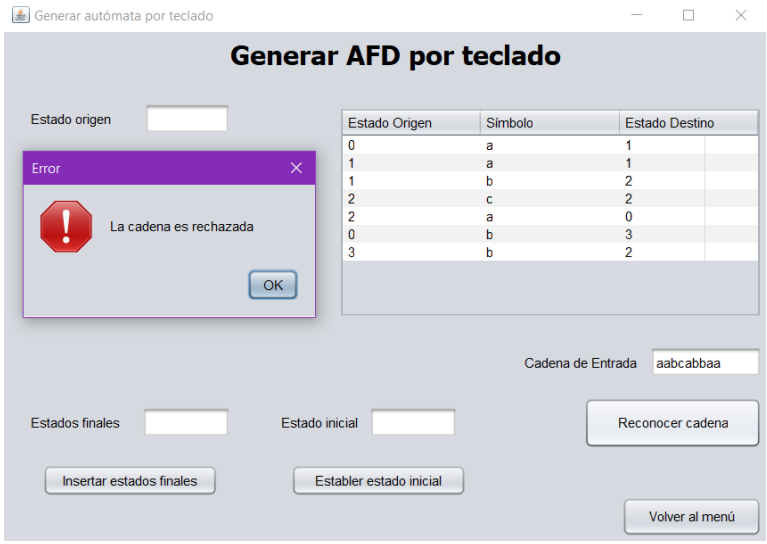
Cadena: abba -> aceptada



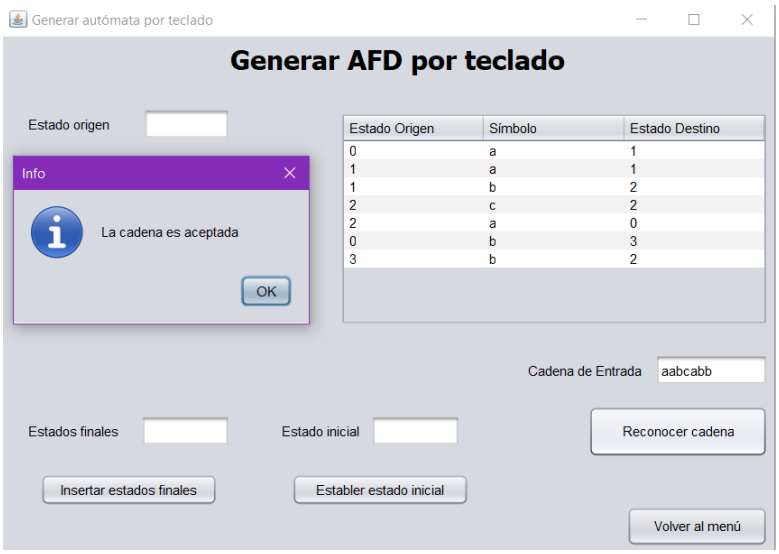
- Ejemplo 2: un grafo un poco más complejo



Cadena: aabcabbaa -> rechazada

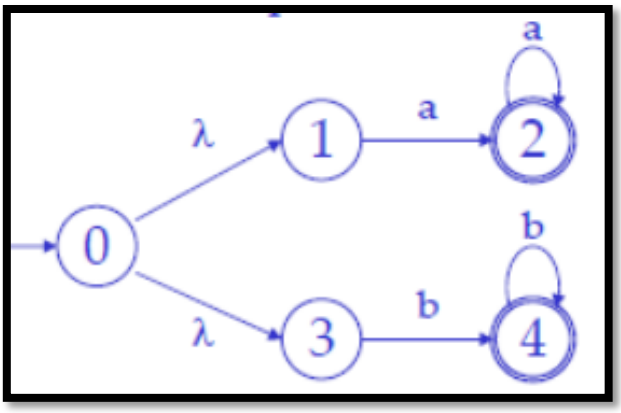


Cadena: aabcabb -> aceptada

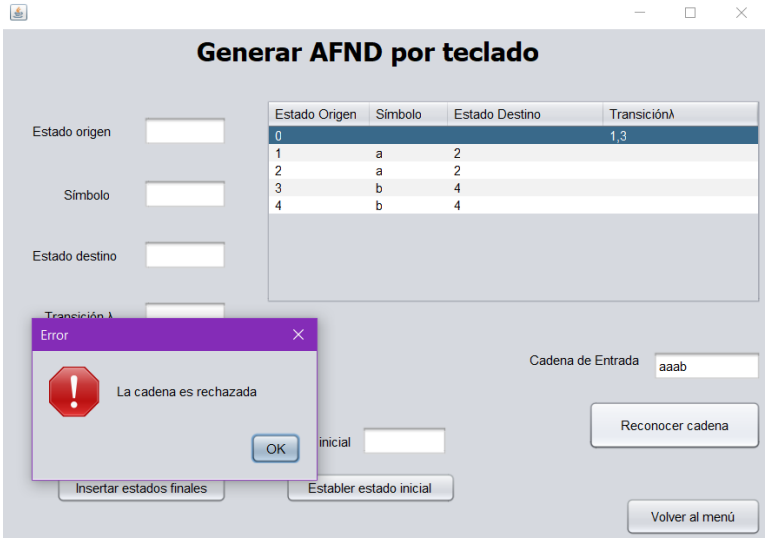


EJEMPLOS AFND

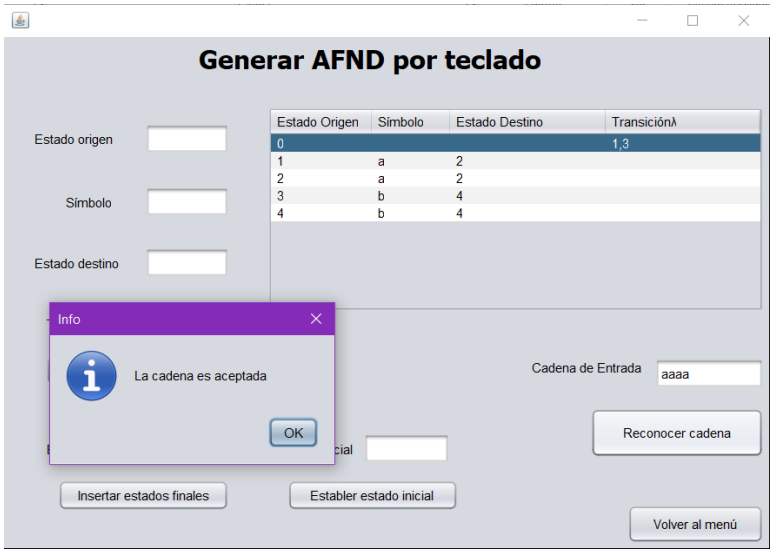
- Ejemplo 1: ejemplo del pdf de la práctica (solo acepta cadenas de aes o bs)



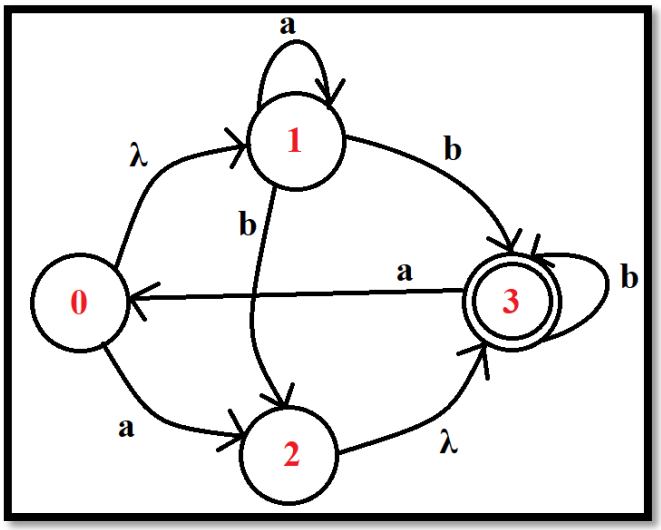
Cadena: aaab -> rechazada



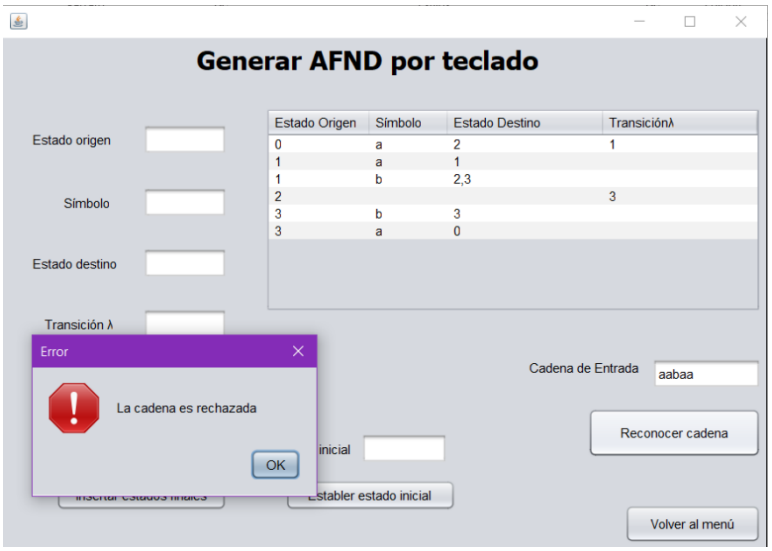
Cadena: aaaa -> aceptada



- Ejemplo 2: uno con más transiciones



Cadena: aabaa -> rechazada



Cadena: aab -> aceptada