



AMD

Grado en Ingeniería Informática

3º Curso

Práctica 2: AFD y AFND.

OBJETIVOS

El objetivo principal de esta práctica es desarrollar un simulador de Autómatas Finitos Determinista (AFD) y de Autómatas Finitos No Determinista (AFND) en Java, **de forma que las transiciones del Autómata que pretendamos simular puedan ser dadas por teclado o leídas desde un fichero.**

Para ello se implementará una aplicación en modo texto (obligatorio) o modo gráfico (opcional) que permita tanto la lectura de un fichero con la especificación del Autómata, como la carga del Autómata desde teclado. La aplicación incluirá un campo para introducir la cadena de entrada a analizar.

Una vez seleccionados el autómata y la cadena de entrada, la aplicación debe simular el comportamiento del autómata, realizando las transiciones correspondientes y mostrando el estado del autómata en cada momento y el resultado final de la simulación (ya sea la aceptación o el rechazo de la cadena).

Autómatas Finitos

Un Autómata Finito es una máquina de estados formada por cinco componentes:

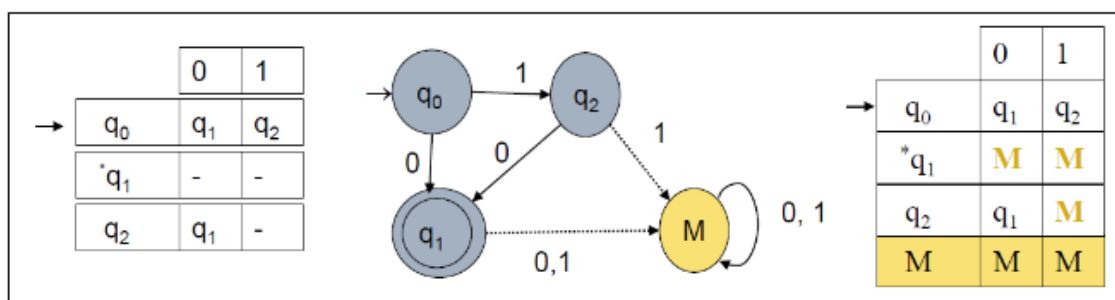
- Un conjunto finito de estados.
- Un conjunto finito de símbolos que pueden ser utilizados como alfabeto de entrada. La entrada del autómata estará formada por una secuencia finita de símbolos del alfabeto.
- Un conjunto finito de transiciones, que están formadas por el estado origen, el estado destino y el símbolo del alfabeto que activa la transición.
- Un estado inicial, perteneciente al conjunto de estados, que describe el estado del autómata antes de comenzar a analizar la cadena de entrada.
- Un conjunto de estados finales que corresponde a un subconjunto de los estados del autómata. La cadena de entrada se aceptará si el autómata queda en un estado final después de las transiciones correspondientes a los símbolos de la cadena de entrada.

El funcionamiento de un Autómata Finito es el siguiente:

- Al comienzo del análisis el autómata se encuentra en el estado inicial.
- El autómata lee de forma secuencial los símbolos de la cadena de entrada. Cada símbolo provoca un cambio de estado del autómata, correspondiente a la transición indicada por el estado actual y el símbolo leído.
- Al terminar el recorrido de la cadena de entrada, si el estado del autómata es un estado final la cadena se ACEPTA. Si el estado no es un estado final, la cadena se RECHAZA.

Los Autómatas Finitos se pueden representar:

- **en forma de tabla:** asignando una fila a cada estado (marcando el estado inicial y los estados finales) y una columna a cada símbolo. Las celdas de la tabla indican el estado destino de las transiciones.
- **en forma diagramática:** cada estado se representa como un círculo y las transiciones se representan como arcos etiquetados. El estado inicial se indica con una flecha y los estados finales se denotan con un círculo doble.



Es frecuente que en la especificación de un autómata no aparezcan todas las transiciones, apareciendo celdas vacías en la representación tabular. Se considera que todas las transiciones no definidas son transiciones a un estado muerto.

Autómatas Finitos Deterministas (AFD) y No Deterministas (AFND)

- Los autómatas descritos anteriormente se denominan Autómatas Finitos Deterministas (AFD) porque dado un estado y un símbolo sólo puede existir una transición (o ninguna si no consideramos el estado muerto). Esto se verifica en la representación tabular viendo que en las celdas solo puede haber un estado (o ninguno si no consideramos el estado muerto).
- Se denominan Autómatas Finitos No Deterministas (AFND) a los autómatas que no verifican la condición anterior, es decir, que pueden tener transiciones con el mismo origen y símbolo que dirijan a diferentes estados de destino. En forma tabular esto quiere decir que en cada celda habría un conjunto de estados de destino. Los AFND pueden tener además transiciones lambda, que son transiciones que no consumen ningún símbolo. Esto quiere decir que el autómata puede cambiar de estado sin leer ningún carácter en la cadena de entrada.

AFND

El funcionamiento de un AFD se puede observar en la siguiente figura:

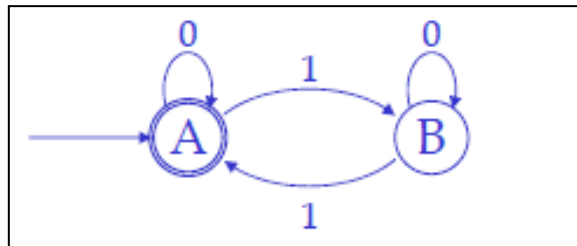


Figura 1: Ejemplo de AFD

En un AFD sólo existe un camino posible, es decir, dada una cadena sólo es posible llegar a un estado a partir del estado inicial, ya que en los AFD dado un estado y un símbolo sólo puede existir una transición (o ninguna si consideramos el estado muerto)

A modo de ejemplo, una posible **implementación simplificada** de un AFD puede ser la siguiente (el alumno puede utilizar las estructuras de datos y clases Java que considere más cómodas y oportunas).

AFD

```
package libClases;

public class AFD {
    private int [ ] estadosFinales; //indica cuales son los estados Finales
    private List<TransicionAFD> transiciones; //indica la lista de transiciones del AFD

    public AFD();
    public agregarTransicion(int e1, char simbolo, int e2);
    public int transicion(int estado, char simbolo);
    public boolean esFinal(int estado);

    public boolean reconocer(String cadena) {
        char[ ] simbolo = cadena.toCharArray();
        int estado = 0 ; //El estado inicial es el 0
        for(int i=0; i<simbolo.length; i++) {
            estado = transicion(estado,simbolo[i]);
        }
        return esFinal(estado);
    }

    public static AFD pedir();
}
```

La clase AFD describe un autómata finito determinista genérico mediante un estado inicial, una lista de transiciones y un conjunto de estados finales. Los estados se representan mediante números enteros, los símbolos mediante caracteres y las transiciones mediante tuplas de 3 elementos: (estado_origen, símbolo, estado_destino).

En esta **implementación simplificada** de un AFD propuesta, los estados del autómata se etiquetan/identifican con un entero¹ y para representar el estado inicial se considera siempre el valor 0, por lo que no ha sido necesario un atributo para indicar cuál es el dicho estado ².

- **agregarTransicion(int e1, char simbolo, int e2):** añade la transición indicada a la lista de transiciones.
- **int transicion(int estado, char simbolo):** dado un estado y un símbolo devuelve el estado destino de la transición o -1 en caso que no exista dicha transición.
- **boolean esFinal(int estado):** devuelve true si estado es un estado final,.
- **boolean reconocer(String cadena):** true si la cadena pertenece al lenguaje descrito en el AFD.
- **static AFD pedir():** Pide por teclado o lee de un fichero (ambas opciones) las transiciones y estados finales del AFD que se desea implementar.

¹ en la versión a implementar por el alumno, los estados se podrán etiquetar/identificar con cualquier literal alfanumérico

² en la versión a implementar por el alumno, el estado inicial podrá ser cualquier otro, por lo que la clase tendrá que tener un atributo específico para indicar cuál es dicho estado

AFND

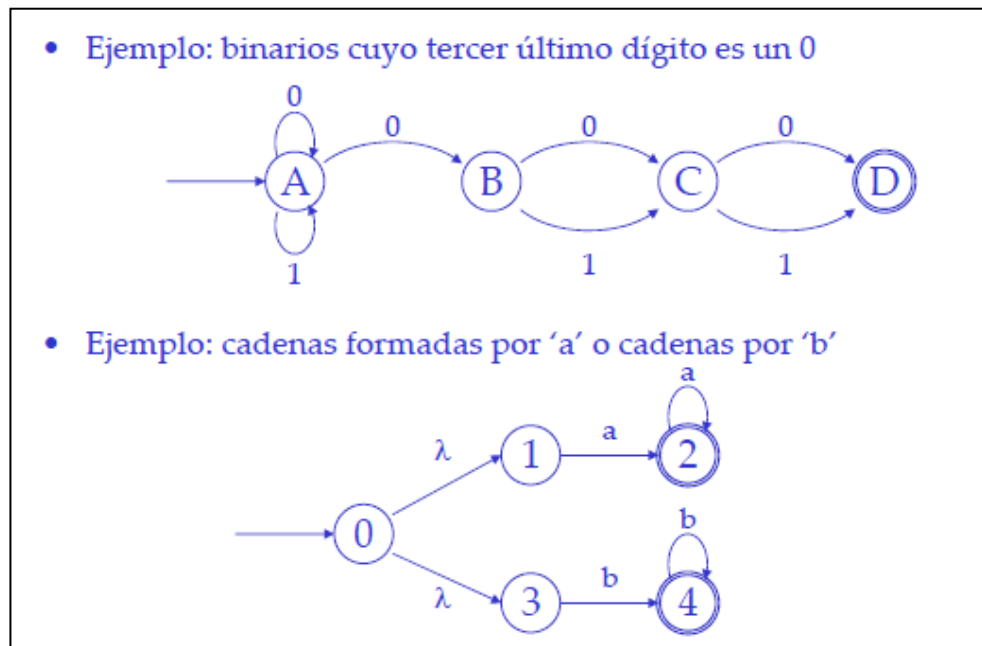
Mientras que en un AFD sólo existe un camino posible, es decir, dada una cadena sólo es posible llegar a un estado a partir del estado inicial (dado un estado y un símbolo sólo puede existir una transición), en un AFND pueden existir varios caminos, de forma que una cadena es aceptada si al menos uno de los caminos conduce a un estado final (puede haber transiciones con el mismo origen y símbolo que dirijan a diferentes estados de destino).

Además, los AFND pueden tener transiciones lambda, que son transiciones que no consumen ningún símbolo. Esto quiere decir que el autómata puede cambiar de estado sin leer ningún carácter en la cadena de entrada.

Funcionamiento de un Autómata Finito No Determinista:

- El funcionamiento de un AFND se basa en que el estado del autómata en un instante se representa como un subconjunto de estados y no como un estado individual.
- Se denomina clausura-lambda de un conjunto de estados al resultado de añadirle a dicho conjunto todos aquellos estados que puedan alcanzarse mediante transiciones lambda.
- Al comienzo del análisis, el Autómata se encuentra en el subconjunto de estados formado por la clausura lambda del estado inicial.
- Las transiciones del Autómata calculan todos los estados accesibles desde los estados del subconjunto origen. El resultado final es la clausura lambda del conjunto de estados construido por las transiciones individuales.
- La cadena de entrada se acepta cuando el subconjunto de estados al terminar el análisis contiene algún estado final.

El funcionamiento de un AFND se puede observar en las siguientes figuras:



Como podemos observar, en un AFND pueden existir varios caminos, de forma que una cadena es aceptada si al menos uno de los caminos conduce a un estado final.

Dado un conjunto de estados E , se denomina λ -clausura(E) al conjunto de estados formado por todos los estados de E más aquellos estados a los que se puede acceder desde E mediante transiciones λ .

El macroestado inicial del AFN es λ -clausura (q_0).

Partiendo de un macroestado E_i , el resultado de una transición con el símbolo 'a' es un macroestado $E_{i+1} = \lambda$ -clausura (E'), donde E' está formado por todos aquellos estados que se puedan alcanzar con transiciones con el símbolo 'a' a partir de alguno de los estados de E_i .

A modo de ejemplo, una posible **implementación simplificada**³ de un AFND puede ser la siguiente:

AFND

```
package libClases;

public class AFND {
    private int [ ] estadosFinales; //indica cuales son los estados Finales
    private List<TransicionAFND> transiciones; //indica la lista de transiciones del AFND
    private List<Transicionλ> transicionesλ; //indica la lista de transiciones λ del AFND

    public AFND();
    public agregarTransicion(int e1, char simbolo, int [ ] e2);
    public agregarTransicionλ(int e1, int [ ] e2);
    private int [ ] transicion(int estado, char simbolo);
    public int [ ] transicion(int [ ] macroestado, char simbolo);
    public int [ ] transicionλ (int estado) ;
    private boolean esFinal(int estado);
    public boolean esFinal(int [ ] macroestado);
    private int[ ] λ_clausura(int[ ] macroestado);

    public boolean reconocer(String cadena) {
        char[ ] simbolo = cadena.toCharArray();
        int [ ] estado = { 0 }; //El estado inicial es el 0
        int[ ] macroestado = λ_clausura(estado);
        for(int i=0; i<simbolo.length; i++) {
            macroestado = transicion(macroestado, simbolo[i]);
        }
        return esFinal(macroestado);
    }

    public static AFND pedir():
}
```

La clase AFND describe un autómata finito no determinista genérico mediante un estado inicial, una lista de transiciones y transiciones-λ y un conjunto de estados finales. Los estados se representan mediante números enteros, los símbolos mediante caracteres, las transiciones mediante tuplas de 3 elementos: (estado_origen, símbolo, estados_destinos[]) y las λ transiciones mediante tuplas de 2 elementos: (estado_origen, estados_destinos[]).

³ el alumno debe implementar una versión más general que permita indicar cuál es el estado inicial y en la que los estados puedan etiquetarse/identificarse con cualquier literal alfanumérico y para ello puede utilizar las estructuras de datos y clases Java que considere más cómodas y oportunas

En un AFND las transiciones pueden consumir un símbolo o no (transiciones λ). El método `transicion()` describe las transiciones del autómatas y genera como salida la lista de estados a los que se puede avanzar consumiendo un símbolo. El método `transicion λ ()` describe las transiciones λ de cada estado, es decir, la lista de estados a los que se puede llegar por medio de transiciones que no consumen ningún símbolo.

Los macroestados del autómatas se representan como listas de estados (`int[]`). El método `λ _clausura()` se utiliza para añadir a un conjunto de estado todos aquellos estados que sean accesibles por medio de transiciones λ .

CONSIDERACIONES A TENER EN CUENTA

Tanto la clase AFD como la clase AFND deben implementar la interfaz Proceso definida a continuación.

```
package libClases;

public interface Proceso {
    public abstract boolean esFinal(int estado); //true si estado es un estado final
    public abstract boolean reconocer(String cadena) ; //true si la cadena es reconocida
    public abstract String toString( ) ; //muestra las transiciones y estados finales
}
```

FICHEROS DE ENTRADA

Para representar los Autómatas Finitos vamos a utilizar ficheros de texto con el formato indicado a continuación:

- El tipo de autómata (AFD o AFND) se describe en una línea que comienza con la cadena "TIPO:" seguida del tipo de autómata a describir AFD (en caso de que sea un Autómata Finito Determinista) o AFND (caso que sea un Autómata Finito No Determinista).
- El conjunto de estados se describe en una línea que comienza con la cadena "ESTADOS:" seguida de la lista de identificadores de estado separados por espacios.
- El estado inicial se describe en una línea que comienza con la cadena "INICIAL:" seguida del identificador del estado inicial.
- El conjunto de estados finales se describe en una línea que comienza con la cadena "FINALES:" seguida de la lista de identificadores de estados finales separados por espacios.
- El conjunto de transiciones se describe en un bloque de texto que comienza con la línea "TRANSICIONES:" y termina con la línea "FIN".
- Cada transición se describe en una línea de texto que comienza con el identificador del estado origen, seguido del símbolo que provoca la transición y del identificador del estado de destino, separados por espacios.
- La descripción de los símbolos es la utilizada comúnmente para representar un literal de tipo carácter. Los símbolos normales se describen entre comillas simples (por ejemplo, 'a'). Para denotar caracteres no imprimibles se utiliza la barra invertida (por ejemplo, el salto de línea se describe como '\n').

A continuación, se muestra un ejemplo de especificación de un AFD.

```
TIPO: AFD
ESTADOS: q0 q1 q2 q3
INICIAL: q0
FINALES: q2 q3
TRANSICIONES:
  q0 '0' q1
  q0 '1' q2
  q1 '1' q3
  q2 '0' q1
FIN
```

En el caso concreto de los AFND el fichero puede tener 2 tipos de transiciones (transiciones que consumen un símbolo y transiciones lambda que no consumen ningún símbolo) con el formato indicado a continuación:

- El conjunto de transiciones se describe en un bloque de texto que comienza con la línea "TRANSICIONES:".
- Cada transición se describe en una línea de texto que comienza con el identificador del estado origen, seguido del símbolo que provoca la transición y los identificadores de los estados destino, separados por espacios.
- El conjunto de transiciones lambda se describe en un bloque de texto que comienza con la línea "TRANSICIONES LAMBDA:".
- Cada transición lambda se describe en una línea de texto que comienza con el identificador del estado origen, seguido de los identificadores de los estados destino, separados por espacios.

A continuación, se muestra un ejemplo de especificación de un AFND.

```
TIPO: AFND
ESTADOS: q0 q1 q2 q3
INICIAL: q0
FINALES: q2 q3
TRANSICIONES:
  q0 '0' q1
  q0 '1' q2 q3
  q1 '1' q0 q2 q3
  q2 '0' q1
TRANSICIONES LAMBDA:
  q0 q1
  q3 q0 q2
FIN
```

DESARROLLO DE LA PRÁCTICA

Como se ha dicho al principio del enunciado, el objetivo principal de esta práctica es desarrollar un proyecto en lenguaje JAVA que simule el funcionamiento de un AFD y un AFND, de forma que las transiciones del Autómata que pretendamos simular puedan ser dadas por teclado o leídas de un fichero (ambas opciones deben ser implementadas).

En caso de seleccionar cargar el autómata desde fichero, el programa debe proporcionar una opción de menú o una ventana de diálogo (en caso que se haga en modo gráfico) que permita indicar la ruta del fichero que desea abrir. En caso de seleccionar cargar el autómata desde teclado el programa debe permitir introducir todos los datos para su creación.

Además de la carga del autómata, el programa deberá incluir un campo de texto para poder introducir la cadena de entrada que se desea analizar/simular.

El programa deberá indicar si la cadena introducida es ACEPTADA – RECHAZADA por el autómata, además de mostrar la información respecto al estado del autómata en cada instante. Para ello, el programa deberá mostrar una opción de menú o botón (modo gráfico) que permita simular el comportamiento del Autómata paso a paso y otra opción de menú o botón (modo gráfico) que permita simular el Autómata de una sola vez.

Opcional (1.5 puntos):

Hacer que la aplicación muestre por pantalla de forma gráfica el autómata y muestre en un cuadro de texto las transiciones que ha realizado para reconocer la cadena introducida.

Opcional (2 puntos):

Hacer que la aplicación muestre por pantalla de forma gráfica las transiciones que el autómata realiza (debe resaltar dichas transiciones) cada vez que pulsamos el botón paso a paso.

DOCUMENTACIÓN A ENTREGAR

La documentación a entregar para esta práctica debe incluir la siguiente información:

1. Código fuente de todas las clases desarrolladas, debidamente comentado.
2. Documentación de todas las clases usadas mediante la herramienta Java-Doc.
3. Memoria explicativa de cómo se ha implementado la práctica, cuáles son los principales problemas encontrados y modo en el que lo han solucionado.
4. Descripción de un conjunto de autómatas de prueba, que cubran todas las posibles situaciones planteadas, y que deberán de adjuntar a la memoria.

LA PRÁCTICA PODRÁ SER REALIZADA EN GRUPO CON UN MÁXIMO DE 2 COMPONENTES (en ese caso ambos alumnos deben subir la documentación y en la portada se debe indicar los nombres de los integrantes del grupo).

La defensa de la práctica y la comprobación de su correcto funcionamiento se realizará el mismo día de entrega de la práctica.