

## PROYECTO II - AFD

**Objetivo:** En este proyecto se aplicará la teoría de Autómata Finitos Determinísticos con representación matricial y el reconocimiento de cuerdas. El programa se desarrollará totalmente en lenguaje Java y usted deberá restringirse a las indicaciones que se presentan en este documento. Al igual que el proyecto anterior, éste es totalmente obligatorio y deberá presentarse en grupos de hasta **2 integrantes**.

**Descripción:** Dentro de los archivos que le proveemos, se encuentra la clase **AFD.java**. En esta clase debe completar los siguientes métodos:

1. El **constructor de la clase** debe recibir como argumento un String, que será la dirección del archivo donde se encuentra la información del afd (más adelante explicaremos el formato de este archivo).
2. El **método `int getTransition(int currentState, char symbol)`**, que deberá devolver el estado al que el afd debe pasar, si se encuentra en el estado `currentState`, y consume el símbolo `symbol`. Este método no solo se usa en el autograder, sino que le será de mucha utilidad para el funcionamiento de su proyecto.
3. El método **`boolean accept(String cuerda)`**, que devuelva `true` o `false`, dependiendo si la cuerda es aceptada o no por el afd.
4. El método **`void main(String[] args)`**, que recibe como primer argumento de entrada del nombre del archivo que contiene la información del afd y como segundo, una bandera (`-i` o `-f`) que indica su modo de ejecución. El funcionamiento de este método será calificado manualmente, así que es libre de implementarlo como desee siempre que siga las especificaciones que se encuentran más adelante.

**Modo de Ejecución:** Su clase principal debe llamarse **AFD.java**. En el método `main`, usted debe recibir como primer argumento, el nombre del archivo que contiene el AFD. El segundo argumento es una bandera `-i` o `-f`, que nos dice el modo de ejecución (interactivo o batch). Si se selecciona el modo batch, entonces deben mandar un tercer parámetro, que será el nombre de un archivo de texto que contenga las cuerdas que se van a evaluar. En general, su programa debe ejecutarse de la siguiente manera:

```
$ java AFD nombre_del_afd (-i|-f) [nombre_del_archivo_de_cuerdas]
```

Si se selecciona la bandera `-i`, entonces su programa, luego de procesar el afd, debe esperar a que el usuario ingrese una cuerda. Luego de haberla leído la, debe desplegar si es aceptada o rechazada por el afd. Para que su programa termine, debe leer una cuerda de longitud 0.

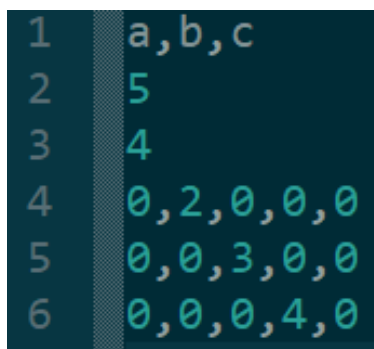
Si se selecciona la bandera `-f`, entonces su programa, luego de procesar el afd, debe leer las cuerdas del archivo especificado en el tercer parámetro. Habrá únicamente una cuerda por línea, y por cada cuerda, al igual que en el modo anterior, su programa debe desplegar si es aceptada o rechazada.

**Formato del archivo:** El archivo del afd será un archivo de texto con extensión “.afd” (por ejemplo **test1.afd**) y éste tendrá el siguiente formato:

- La primera línea contendrá los símbolos terminales (o el alfabeto de entrada) separados por una “coma”. Tome en cuenta que el alfabeto de entrada es un subconjunto de los caracteres ASCII den 33 al 127, excluyendo la coma (“El carácter ‘\n’ no pertenece a este conjunto”).
- La segunda línea contendrá la cantidad de estados del afd (incluido el estado absorbente).
- La tercera línea contendrá los estados finales separados por una “coma”.
- Las siguientes líneas contendrán las filas de la matriz de transición, es decir, contendrán una secuencia de estados separados por comas. Cada fila, tendrá N estados, donde N es el número de estados totales del afd, y habrá en total M filas, una fila por cada símbolo del alfabeto.

Se utilizarán números enteros para nombrar a los estados del AFD. El estado inicial será reconocido por el número “1” y el estado absorbente por el número “0”.

Veamos un Ejemplo:



1	a,b,c
2	5
3	4
4	0,2,0,0,0
5	0,0,3,0,0
6	0,0,0,4,0

La primera línea, especifica el alfabeto de entrada como  $S=\{a,b,c\}$

La segunda línea nos indica que habrá 5 estados

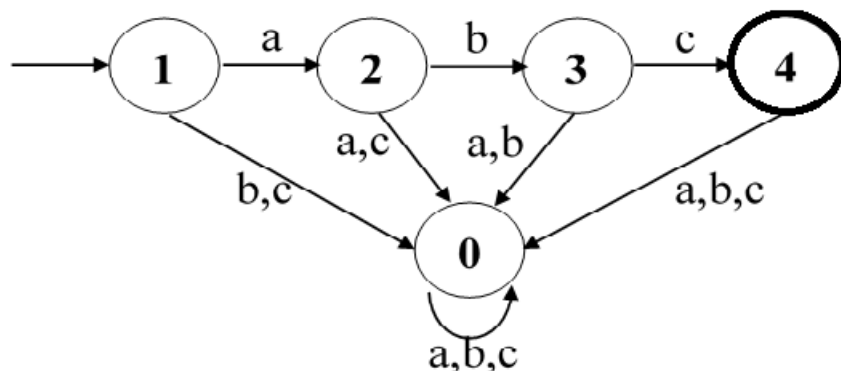
La tercera línea dice que el estado final es el 4.

La cuarta línea son las transiciones con el símbolo ‘a’, Si se está en el estado 0, debe quedarse en el 0, si está en el 1, debe pasar al 2, si esta el 2, 3, o 4, debe cambiar al estado 0.

La quinta línea son las transiciones con el símbolo ‘b’. Si se encuentra en el estado 0, 1,3 o 4, debe cambiar al 0, y si está en el estado 2, debe cambiar al 3.

La sexta línea son las transiciones con el símbolo ‘c’. Si se encuentra en el estado 0, 1,2 o 4, debe cambiar al 0, y si está en el estado 3, debe cambiar al 4.

Traduciendo el AFD a la forma gráfica, se vería así:



Y en forma matricial:

Estado/ Símbolo	a	b	c
0	0	0	0
1	2	0	0
2	0	3	0
3	0	0	4
4	0	0	0

### Indicaciones Extra:

1. Si su programa no compila, tendrán una nota automática de 0.
2. Tome en cuenta que todos los afd y cuerdas que proporcionaremos al momento de calificar son válidos, así que no se deben preocupar tanto por las verificaciones en este caso.
3. Como el modo de ejecución **-i** debe terminar al encontrar una cuerda de longitud 0, ningún AFD usado para calificar tendrá como estado inicial el estado final.
4. Si el archivo. afd que usted ingresa al autograder no es válido, es probable que funcione, y que devuelva siempre “rechazado”, lo mismo pasará cuando haya una cuerda que contenga un símbolo no especificado en el alfabeto de entrada.

### Material de Apoyo y Autograder:

En el GES, en la sección Material de Apoyo, ustedes encontraran una carpeta llamada [pj2.zip](#). La carpeta contiene una la clase AFD.java que deben implementar, una serie de afd válidos (en la subcarpeta tests/afd) y unas cuerdas de prueba para cada afd (en la subcarpeta tests/cuerdas). Una vez terminen su proyecto, pueden ejecutar el autograder de la siguiente manera:

## \$ java Grading -afd nombre\_del\_afd nombre\_del\_archivo\_de\_cuerdas

Al ejecutarse, Grading creará dos afd, uno usando la clase que usted implementó, y uno utilizando una clase que nosotros implementamos, y comparará las respuestas que obtengan los dos. De esta forma usted puede crear un archivo. afd y un archivo de cuerdas, y comprobar si su proyecto funciona correctamente. Es más, **para este proyecto lo alentamos a que cree sus propios archivos de prueba y asegurarse que su afd funciona de manera adecuada.** Esto es porque la calificación utilizará diferentes tests de los que nosotros le proveemos, y serán mucho más largos y exhaustivos.

Veamos un ejemplo de la ejecución:

```
volant360 solucion $ java Grading -afd tests/afd/hex.afd tests/cuerdas/hex.txt
Test 1: Creacion de un AFD..... (+5)
Test 2: Probando Transiciones Aleatorias..... (+1)
Probando D(2,B). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(1,E). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(2,D). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(3,F). Resultado Obtenido: 4, Resultado Esperado: 4 .....(+1)
Probando D(2,1). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(3,C). Resultado Obtenido: 4, Resultado Esperado: 4 .....(+1)
Probando D(3,A). Resultado Obtenido: 4, Resultado Esperado: 4 .....(+1)
Probando D(2,7). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(3,6). Resultado Obtenido: 4, Resultado Esperado: 4 .....(+1)
Probando D(3,5). Resultado Obtenido: 4, Resultado Esperado: 4 .....(+1)
Test 2 Terminado.....
Test 3: Probando Cuedas (El Archivo Contiene 6 Cuerdas).....
Probando ' 0x099'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.17)
Probando ' x932'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Probando ' 0x128'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.17)
Probando ' 1x33456323234'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Probando ' 0x9900930001111'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.17)
Probando ' 0xx'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Grading Terminado. Nota Final: ..... 100/100
volant360 solucion $
```

Algo muy importante a tomar en cuenta es que, al momento de aceptar una cuerda, pueden pasar dos cosas, que su afd la acepte o la rechace. Como solo hay dos opciones, la probabilidad de acertar, al intentar adivinar el resultado es 50%. Aquí hay un ejemplo, en el que corremos el autograder con los archivos base que les proporcionamos sin ninguna modificación. Tomen en cuenta que, para la calificación, utilizaremos un autograder que verifique este tipo de problemas:

```
volant360 alumno $ java Grading -afd tests/afd/hex.afd tests/cuerdas/hex.txt
Test 1: Creacion de un AFD..... (+5)
Test 2: Probando Transiciones Aleatorias..... (+1)
Probando D(2,6). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(3,9). Resultado Obtenido: 0, Resultado Esperado: 4 .....(+0)
Probando D(4,6). Resultado Obtenido: 0, Resultado Esperado: 4 .....(+0)
Probando D(3,x). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(2,9). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(3,A). Resultado Obtenido: 0, Resultado Esperado: 4 .....(+0)
Probando D(2,3). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Probando D(4,0). Resultado Obtenido: 0, Resultado Esperado: 4 .....(+0)
Probando D(3,8). Resultado Obtenido: 0, Resultado Esperado: 4 .....(+0)
Probando D(1,6). Resultado Obtenido: 0, Resultado Esperado: 0 .....(+1)
Test 2 Terminado.....
Test 3: Probando Cuedas (El Archivo Contiene 6 Cuerdas).....
Probando ' 0x099'. Resultado Obtenido: false, Resultado Esperado: true.... (+ 0.00)
Probando ' x932'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Probando ' 0x128'. Resultado Obtenido: false, Resultado Esperado: true.... (+ 0.00)
Probando ' 1x33456323234'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Probando ' 0x9900930001111'. Resultado Obtenido: false, Resultado Esperado: true.... (+ 0.00)
Probando ' 0xx'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.17)
Grading Terminado. Nota Final: ..... 53/100
volant360 alumno $
```

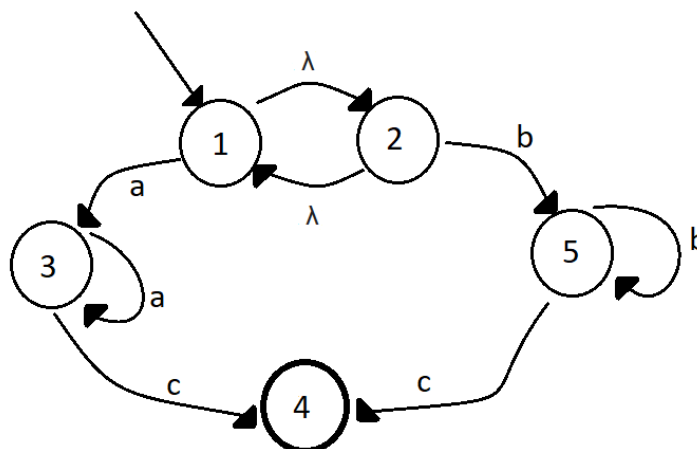
### Puntos Extra (+50 puntos extra):

Si usted desea un reto mayor al del proyecto del AFD, en el que aprenderá mucho más, y se sentirá realizado cuando lo logre (porque si confiamos en que pueden hacerlo), esto es lo que le proponemos: que deje de lado la clase AFD.java e implemente la clase AFN.java. La base del proyecto seguirá siendo la misma, con unas cuantas modificaciones:

1. Ya no deberá implementar el método **int getTransition(int currentState, char symbol)** porque en un AFD, la transición no es un estado específico, sino un conjunto de estados posibles.
2. El formato del archivo “.afd” cambiará ligeramente, Ahora la matriz de transición tendrá una fila extra al inicio, que representará las transiciones hechas con lambda. También, como se las transiciones se hacen a un conjunto de estados, y no a un estado en particular, en la matriz de transición, en vez de tener un solo número, tendrá uno o más números separados por “;”, que será este conjunto de estados a los que se puede ir. Veamos un ejemplo:

```
1 | a,b,c
2 | 6
3 | 4
4 | 0,1;2,2;1,3,4,5
5 | 0,3,0,3,0,0
6 | 0,0,5,0,0,5
7 | 0,0,0,4,0,4
```

La línea número 4 nos dice que, con lambda, del estado 0, podemos llegar al estado 0, del estado 1, podemos llegar a los estados 1 y 2, del estado 2, podemos llegar a los estados 2 y 1, y de los estados 3, 4 y 5 podemos llegar únicamente a ellos mismos. En este ejemplo, las demás líneas solo tienen una única transición, pero podrían tener más, y su AFN deberá poder interpretarlo correctamente. Esta sería la forma gráfica del AFN que anterior:



3. Al igual que con AFD.java, usted debe implementar el método main(), que podrá ejecutar en modo batch y modo interactivo.
4. El autograder se ejecuta de la siguiente manera (una vez más, **lo alentamos a que usted haga sus propios tests**, porque los que nosotros le proveemos son simples, pero le damos las herramientas para probar cualquier afn que desee):

**\$ java Grading -afn nombre\_del\_afn nombre\_del\_archivo\_de\_cuerdas**

```

volant360 solucion $ java Grading -afn tests/afn/multiple_transitions.afn tests/cuerdas/multiple_transitions.txt
Test 1: Creacion de un AFN..... (+5)
Test 2: Probando Cuedas (El Archivo Contiene 10 Cuerdas).....
Probando ' 0'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.50)
Probando ' 101'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Probando ' 0001'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Probando ' 00010'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.50)
Probando ' 10'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.50)
Probando ' 1000000'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Probando ' 000000001'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Probando ' 0010'. Resultado Obtenido: true, Resultado Esperado: true.... (+14.50)
Probando ' 00100'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Probando ' 1'. Resultado Obtenido: false, Resultado Esperado: false.... (+14.50)
Grading Terminado. Nota Final: ..... 150/100
volant360 solucion $

```

**Fecha de Entrega: 4 Abril de 2021.**

**Método de Entrega:** El proyecto se realizará en grupos **de máximo dos estudiantes** de la misma sección. La entrega se realizará mediante el GES en la asignación respectiva (UNICAMENTE un integrante del grupo debe subir el proyecto). Deben crear una carpeta llamada PJ2\_CARNET1\_CARNET2 (ejemplo: pj2\_10000000\_10000001) y dentro de ella colocar únicamente los archivos.java que utilizaron. Luego compriman la carpeta en formato .zip y subirla al GES. Deben asegurarse de que todos los archivos compilan correctamente. **Si no compila automáticamente tienen cero en el proyecto.**

**Recuerde el grupo para el proyecto 2, debe de ser el mismo para el proyecto 3.**

#### **Notas Finales:**

- La forma más fácil de resolver este problema es con recursión sin embargo el proyecto **no restringe el uso de ciclos**.

**Copia:**

- En caso de copia con estudiantes de este o cualquier otro año, se recurrirá al Reglamento de Universidad Galileo. Artículo 71.1 en la primera incidencia o segunda incidencia 71.2.