21-1-2020

Memoria de la práctica 3

AFD y AFND

Alberto de la torre solis

ALBERTO.DELATORRE117@ALU.UHU.ES

CONTENIDO

ILUSTRACIONES

LEER DE FICHERO

Para leer los ficheros ha sido necesario crear una clase llamada “Lector” en la cual he creado una función llamada “readFile” la cual se encargar de realizar dicha función. La sintaxis que tiene que seguir un fichero que creemos para ser leído debe ser de la siguiente forma:

```
#AFD
#Estados A B D
#Finales C
A,0,A
A,1,B
B,0,B
B,1,C
C,0,C
C,1,D
D,1,B
D,0,A
#EOF
```

Ilustración 2: Definición AFD

```
#AFND
#Estados A B D
#Finales C
A,0,A
A,1,B
B,0,B
B,1,C
C,0,C
C,1,D
D,1,B
D,0,A
#EOF
```

Ilustración 1: Definición AFND

A la hora de definir un estado a través de fichero, este solo podrá estar formado por un solo carácter.

El funcionamiento de la función readFile será el de comprobar que tipo de autómata contiene el fichero y crear un “Proceso” AFD o AFND, con ello podremos tener el autómata necesario para guardar dentro el contenido del fichero.

Para acceder a esta funcionalidad mediante la aplicación, se podrá acceder desde una pestaña en la parte superior Nuevo > Leer de Fichero.

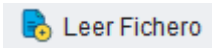

Ilustración 3: Leer Fichero

Se han definido 5 ficheros que recogen todos los posibles autómatas que pueden definirse, tanto AFD como AFND. A continuación, se mostrará el código, junto con una breve explicación de este, y una imagen que representa el autómata que hemos creado.

AFD

Como se puede observar en el recuadro inferior, hemos definido un autómata AFD con 4 estados, A, B, C y D. A continuación, se han definido las distintas transiciones entre estos.

```
#AFD
#Estados A B D
#Finales C
A,0,A
A,1,B
B,0,B
B,1,C
C,0,C
C,1,D
D,1,B
D,0,A
#EOF
```

Ilustración 4: AFD ejemplo txt

El autómata resultante se identifica con la siguiente imagen que se ha generado con el programa Graphviz:

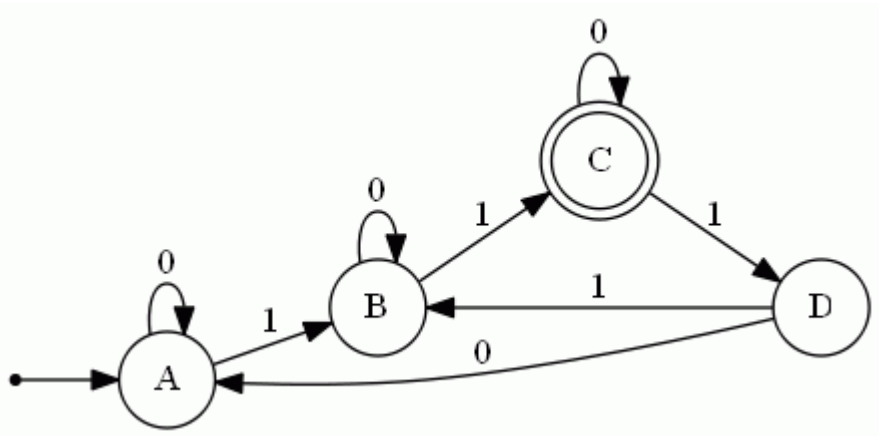

Ilustración 5: AFD ejemplo gráfico

AFND SIMPLE

Para abarcar todas las opciones posibles he definido un autómata AFND iguál al AFD anterior con los mismos estados y transiciones.

```
1   #AFND
2   #Estados A B D
3   #Finales C
4   A,0,A
5   A,1,B
6   B,0,B
7   B,1,C
8   C,0,C
9   C,1,D
10  D,1,B
11  D,0,A
12  #EOF
```

Ilustración 6: AFND simple ejemplo txt

El autómata resultante se identifica con la siguiente imagen que se ha generado con el programa Graphviz:

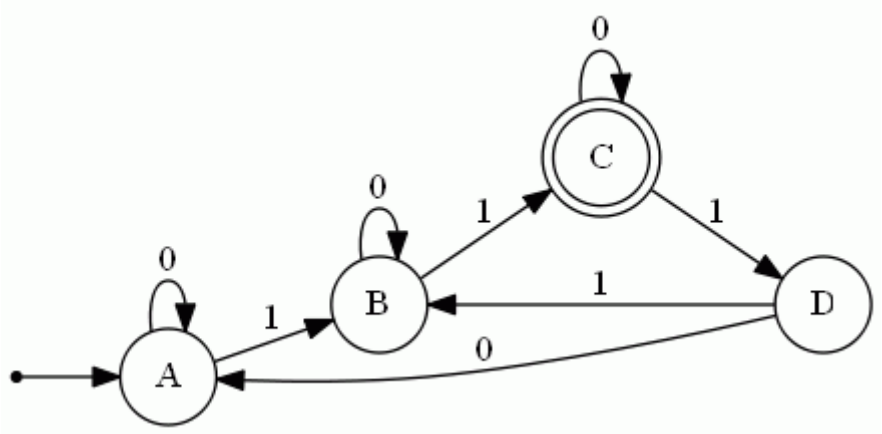

Ilustración 7: AFND simple ejemplo gráfico

AFND SIN TRANSICIONES Λ

En este caso he definido un autómata finito no determinista peros sin definir transiciones λ. Para no complicar el ejemplo definimos 4 estados A, B, C y D y las distintas transiciones que se dan entre estos. Como se puede observar hay estados que con el mismo simbolo se puede llegar a estados distintos.

```
1   #AFND
2   #Estados A B
3   #Finales C
4   A,0,A
5   A,1,B
6   A,1,C
7   B,1,B
8   B,0,C
9   C,0,C
10  C,1,A
11  C,0,B
12  #EOF
```

Ilustración 8: AFND sin transiciones λ ejemplo txt

El autómata resultante se identifica con la siguiente imagen que se ha generado con el programa Graphviz:

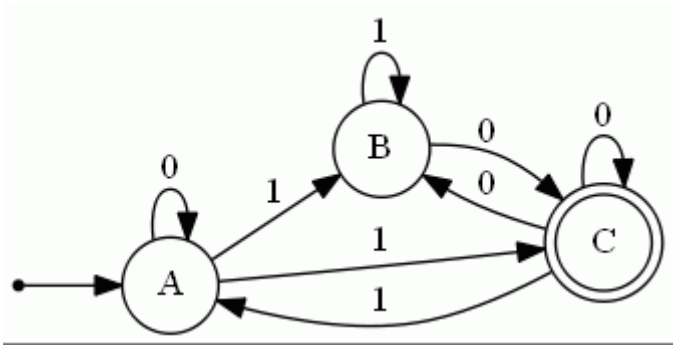

Ilustración 9: AFND sin transiciones λ ejemplo gráfico

AFND CON TRANSICIONES Λ

En este último ejemplo hemos definido un AFND con transiciones λ. Hemos vuelto a definir 4 estados y las transiciones que se dan entre estos. Además, se han definido transiciones λ como puede verse en las transiciones que parten desde el estado A al estado B y C.

```
1   #AFND
2   #Estados A B
3   #Finales C
4   A,0,A
5   A,l,B
6   A,l,C
7   B,1,B
8   B,0,C
9   C,0,C
10  C,1,A
11  C,0,B
12  #EOF
```

Ilustración 10: AFND con transiciones λ ejemplo txt

El autómata resultante se identifica con la siguiente imagen que se ha generado con el programa Graphviz:

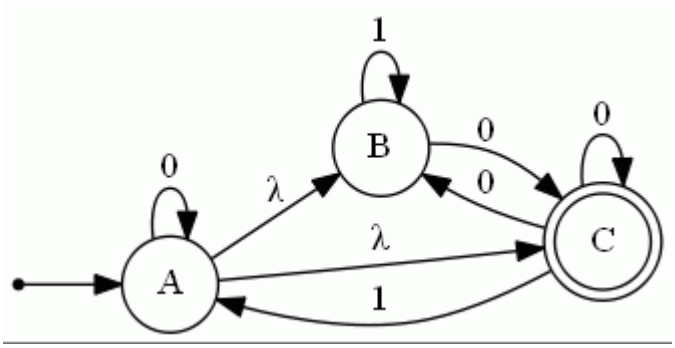

Ilustración 11: AFND con transiciones λ ejemplo gráfico

RECONOCIMIENTO DE CADENAS

Una de las finalidades de la práctica es introducir una cadena de símbolos, y que se compruebe si el autómata acepta o no esa cadena. A esta opción se puede acceder mediante un botón.

Ilustración 12: Reconocer Cadena

Para mostrar el funcionamiento, utilizaré un autómata AFD y compruebo varias opciones. El autómata que nos servirá de ejemplo es el siguiente:

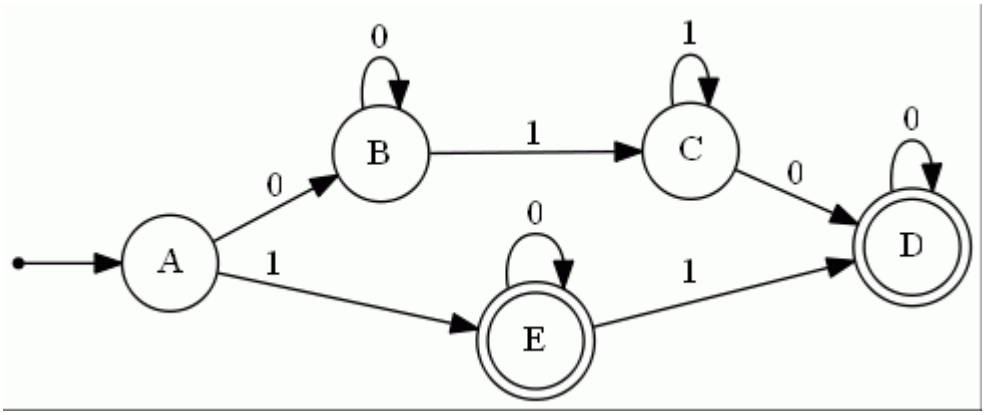

Ilustración 13: AFD ejemplo

Siempre que ejecutemos una cadena, cuyo carácter final termine en un estado final, la cadena nos la dará por buena. De este modo aceptará cadenas como: 1, 10, 101, 0010, 00110, 001100.

Ahora realizaré las comprobaciones con un autómata AFND que será el siguiente:

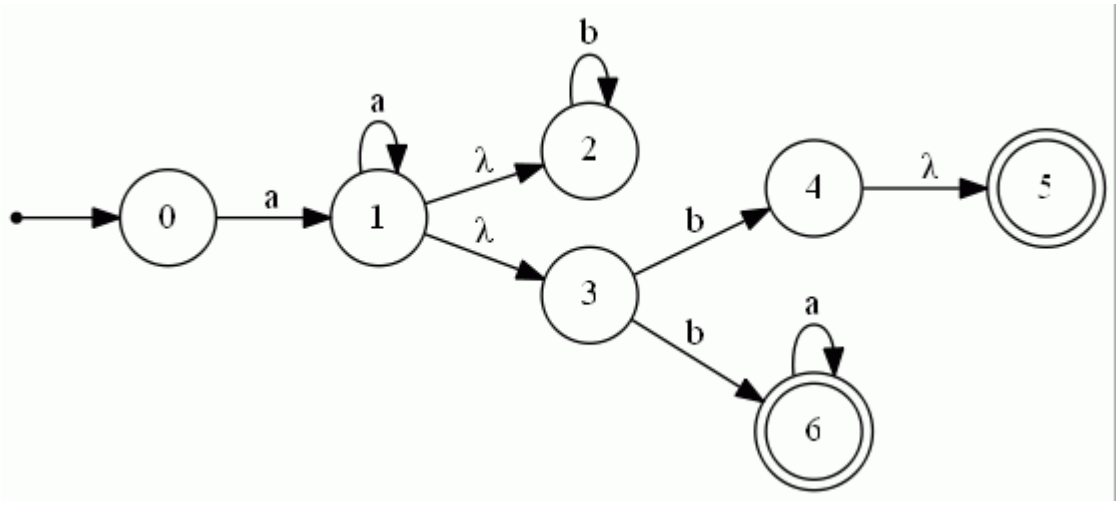

Ilustración 14: AFND ejemplo

En este caso tendremos que tener en consideración las transiciones λ, pues una vez lleguemos a un estado que tenga alguna, el estado al que apunta esta, pertenecerá también al macroestado. De este modo vamos a ver una serie de cadenas de ejemplo.

Voy a poner un ejemplo sencillo, insertamos la cadena “ab”. Obtendremos el mensaje “La cadena ab es aceptada.”, además de obtener los distintos macroestados a través de los cuales pasa la secuencia de símbolos, que son los siguientes:

Macroestado 0 [Estado: 0]

Macroestado 1 [Estado: 1, Estado: 2, Estado: 3]

Macroestado 2 [Estado: 3, Estado: 6, Estado: 4, Estado: 5]

Esto significa que, en el momento inicial, estamos ante el macroestado 0, el cual está formado por el estado 0. Al llegar el símbolo a, llegamos al estado 1, que está formado por este mismo, más todos los estados a los cuales se llega mediante una transición λ, de este modo el macroestado 1 estará formado por los estados 1, 2 y 3. Por último, al llegar el símbolo b, desde el estado 3, ambas transiciones desembocarían en estados finales, es por ello que ambos estados, 5 y 6, forman parte del macroestado 2, estando este compuesto por los estados 3, 4, 5 y 6.

Si por ejemplo probáramos con la cadena “aa”, el mensaje que obtendríamos sería el siguiente “La cadena aa no es aceptada. “, esto se debe, a cómo puede comprobarse en la representación del autómata, la cadena terminaría en el macroestado 1, estando este formado por los estados 1, 2 y 3, y no siendo ninguno de ellos estados finales.

Otras cadenas que el autómata aceptaría como válidas serán las siguientes: aab, aba, aaba.

Para hacer las pertinentes comprobaciones, se adjuntan dos ficheros junto a la documentación de la práctica, con los que se cargarían los autómatas que se han representado anteriormente. Estos ficheros están contenidos en la carpeta “Autómatas”, y se llaman “AFD-2” y “AFND-4”.

MÉTODOS RECONOCER Y ΛCLAUSURA

MÉTODO RECONOCER (AFND)

El método reconocer del AFND funciona recibiendo una cadena de tipo String como parámetro.

Se aplica λclausura(Einicial) donde Einicial es el estado inicial del autómata. Posteriormente, se realiza un bucle “for” cuya condición de salida es haber consumido todos los símbolos de la cadena que se pretende procesar.

Este bucle, va obteniendo diferentes macroestados dependiendo de las transiciones que sea capaz de realizar el autómata.

Al finalizar el bucle, si algunos de los estados que forman parte del macroestado es final, la cadena será reconocida por el autómata mientras que si ninguno de los estados es final el autómata determinará que no reconoce dicha cadena.

MÉTODO ΛCLAUSURA

El método λclausura ha sido implementado de manera recursiva. Siguiendo el modelo de implementación de recorrido de grafos sin realizar/evitando ciclos aprendido a lo largo de la titulación.

Tendremos dos métodos, uno que recibe el estado de partida y otro que se encarga de la recursividad. El primer método, crea un macroestado vacío y un diccionario formado por todos los estados del autómata y con el valor 'false' (esta inicialización sirve para indicar que no se ha visitado ningún estado). En la teoría de diccionarios, se distinguen la clave y el valor, par(clave,valor) en nuestro caso, la clave serán los estados y el valor será de tipo booleano e indicará si el estado ha sido visitado o no.

Una vez realizadas estas inicializaciones se realiza la llamada al método recursivo.

λclausura recursivo está parametrizado con el estado y macroestado actual y el diccionario de visitados. En la primera llamada, el estado actual será sobre el que se realizó la llamada al primer método y el macroestado será un macroestado vacío.

Los casos base del método son dos: a) que no haya transiciones λ desde el estado actual o b) que por el contrario estas sí existan.

Caso a): si el estado carece de transiciones λ, se añade el estado actual al macroestado, se marca como visitado y se devuelve el macroestado.

Caso b): si el estado tiene transiciones λ, se añade el estado actual al macroestado, se marca como visitado y se recorren los estados destino de las transiciones λ del estado.

En este recorrido, se selecciona como estado siguiente el primer estado destino de la primera transición y si no ha sido visitado ni está en el macroestado actual se realiza la llamada recursiva con el estado siguiente, el diccionario de visitados y el macroestado actual. Este proceso se hará hasta terminar la lista de transiciones λ seleccionando en cada momento el estado siguiente correspondiente.

Finalmente, el método recursivo devuelve el macroestado resultante de aplicar λclausura al estado de partida con el que se comienza el primer método.

REPRESENTACIÓN DE AUTÓMATAS CON GRAPHVIZ

Graphviz es un paquete de herramientas de código abierto desarrollado por "AT&T Labs Research" que permite crear gráficos a partir de scripts. Además, viene provisto de herramientas y librerías que facilitan su integración con otros programas.

La descarga de Graphviz es gratuita, se puede hacer a través del enlace disponible en su web, http://www.graphviz.org/Download..php. Además, se trata de una aplicación multiplatafoma.

Graphviz es bastante potente, y es capaz de graficar multitud de diagramas distintos, pero a nosotros nos interesa la representación de autómatas. Para este caso hemos encontrado bastantes ejemplos y manuales a través de Internet. Uno que nos sirvió para empezar a comprender la sintaxis fue el que se incluye en la misma web del programa, http://www.graphviz.org/pdf/dotguide.pdf, pero el que de verdad nos ayudó a la hora de definir los autómatas, fue el manual que se incluye en la web de Martin Thoma https://martin-thoma.com/how-to-draw-a-finite-state-machine/, ya que explica de forma concreta la creación de distintos autómatas.

Un ejemplo de codificación de autómata sería el siguiente:

```
digraph finite_state_machine {
  rankdir=LR;
  size="8,5"
  node [shape = point]; start;
  node [shape = circle, color=red] A;
  node [shape = circle] B;
  node [shape = circle] C;
  node [shape = doublecircle, color=blue, fontcolor=black]; D;
  node [shape = doublecircle, color=blue, fontcolor=black]; E;
  A -> B [label = "0"];
  A -> E [label = "1"];
  B -> B [label = "0"];
  B -> C [label = "1"];
  C -> D [label = "0"];
  C -> C [label = "1"];
  D -> D [label = "0"];
  E -> E [label = "0"];
  E -> D [label = "1"];
  start -> A;
}
```

Ilustración 15: Ejemplo Graphviz txt

El autómata resultante sería el que podemos ver en la siguiente imagen:

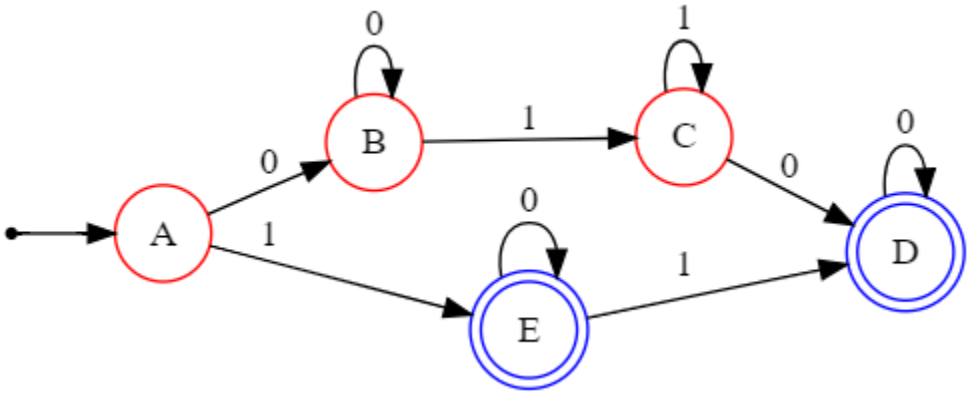

Ilustración 16: Autómata Resultante

Como podemos ver, gracias a una sintaxis bastante sencilla podemos obtener resultados de bastante calidad y que definen con total exactitud el autómata que necesitemos representar.

DIFICULTADES A LA HORA DE DESARROLLAR LA PRÁCTICA

Los problemas que he encontrado a la hora del desarrollo han sido al principio al intentar hacer los estados y los diferentes autómatas ya que no tomé en cuenta que deberían de ser dos tipos diferentes para que a la hora de llevarlo a la practica no hubiera problemas, la solución fue simplemente hacer un segundo estado y así no tenía los problemas que tuve.

Otro problema fue a la hora gráfica con la implementación del graphviz ya que en mi ordenador tuve problemas para que se ejecutara de forma normal por lo que tuve que meter la carpeta del graphviz dentro de mi propia carpeta para que funcionase sin problemas, pero esto le aumenta mucho el peso a la propia carpeta del proyecto.