

Prácticas de Autómatas y Lenguajes. Curso 2018/19

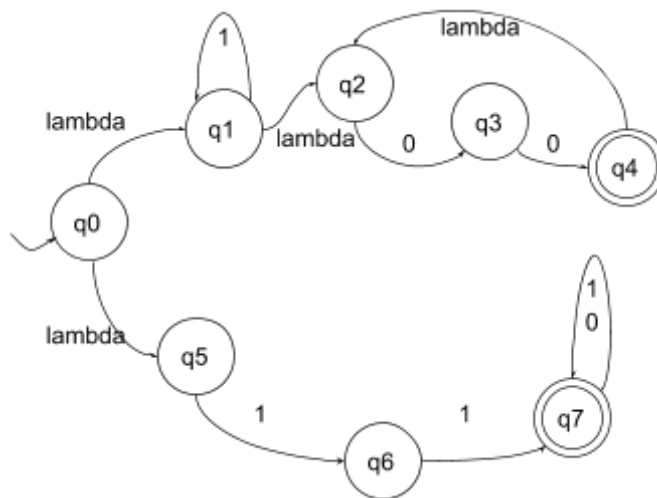
Práctica 2: Simulación de autómatas finitos no deterministas II (transiciones λ)

Descripción del enunciado

El objetivo de esta práctica es que amplíes la librería C de la práctica 1 de manera que los autómatas finitos no deterministas simulados admitan también transiciones λ .

Así, por ejemplo, tu librería tendría que ser capaz de simular el comportamiento del siguiente AFND que podría reconocer el lenguaje representado por la expresión regular

$$[1^*00(00)^*] + [11(0+1)^*]$$



Observa el siguiente posible programa principal para definir mediante tu librería este autómata (las nuevas funciones están resaltadas)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "alfabeto.h"
#include "estado.h"
#include "afnd.h"
#include "palabra.h"

int main(int argc, char ** argv)
{
    AFND * p_afnd_l;
```

```

p_afnd_1 = AFNDNuevo("af11",8,2);

AFNDInsertaSimbolo(p_afnd_1,"0");
AFNDInsertaSimbolo(p_afnd_1,"1");

AFNDInsertaEstado(p_afnd_1,"q0",INICIAL);
AFNDInsertaEstado(p_afnd_1,"q1",NORMAL);
AFNDInsertaEstado(p_afnd_1,"q2",NORMAL);
AFNDInsertaEstado(p_afnd_1,"q3",NORMAL);
AFNDInsertaEstado(p_afnd_1,"q4",FINAL);
AFNDInsertaEstado(p_afnd_1,"q5",NORMAL);
AFNDInsertaEstado(p_afnd_1,"q6",NORMAL);
AFNDInsertaEstado(p_afnd_1,"q7",FINAL);

AFNDInsertaTransicion(p_afnd_1, "q1", "1", "q1");
AFNDInsertaTransicion(p_afnd_1, "q2", "0", "q3");
AFNDInsertaTransicion(p_afnd_1, "q3", "0", "q4");
AFNDInsertaTransicion(p_afnd_1, "q5", "1", "q6");
AFNDInsertaTransicion(p_afnd_1, "q6", "1", "q7");
AFNDInsertaTransicion(p_afnd_1, "q7", "0", "q7");
AFNDInsertaTransicion(p_afnd_1, "q7", "1", "q7");

AFNDInsertaTransicion(p_afnd_1, "q0", "q1");
AFNDInsertaTransicion(p_afnd_1, "q0", "q5");
AFNDInsertaTransicion(p_afnd_1, "q1", "q2");
AFNDInsertaTransicion(p_afnd_1, "q4", "q2");

AFNDCierraTransicion(p_afnd_1);

AFNDImprime(stdout,p_afnd_1);

AFNDInsertaLetra(p_afnd_1,"1");
AFNDInsertaLetra(p_afnd_1,"1");
AFNDInsertaLetra(p_afnd_1,"1");
AFNDInsertaLetra(p_afnd_1,"1");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");

p_afnd_1 = AFNDInicializaEstado(p_afnd_1);

fprintf(stdout,"\n***** PROCESA CADENA *****\n");
AFNDImprimeCadenaActual(stdout,p_afnd_1);
fprintf(stdout,"\n*****\n");

AFNDProcesaEntrada(stdout,p_afnd_1);

/*****
*/

p_afnd_1 = AFNDInicializaCadenaActual(p_afnd_1);
AFNDInsertaLetra(p_afnd_1,"1");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");
AFNDInsertaLetra(p_afnd_1,"0");

p_afnd_1 = AFNDInicializaEstado(p_afnd_1);

```

```

    fprintf(stdout, "\n***** PROCESA CADENA *****\n");
    AFNDProcesaEntrada(stdout, p_afnd_1);
    fprintf(stdout, "\n*****\n");

/*****
*/

    p_afnd_1 = AFNDInicializaCadenaActual(p_afnd_1):
    AFNDInsertaLetra(p_afnd_1, "1");
    AFNDInsertaLetra(p_afnd_1, "1");
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "0");

    p_afnd_1 = AFNDInicializaEstado(p_afnd_1);

    fprintf(stdout, "\n***** PROCESA CADENA *****\n");
    AFNDProcesaEntrada(stdout, p_afnd_1);
    fprintf(stdout, "\n*****\n");

/*****
*/

    p_afnd_1 = AFNDInicializaCadenaActual(p_afnd_1):
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "1");
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "1");
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "1");
    AFNDInsertaLetra(p_afnd_1, "0");
    AFNDInsertaLetra(p_afnd_1, "1");

    p_afnd_1 = AFNDInicializaEstado(p_afnd_1);

    fprintf(stdout, "\n***** PROCESA CADENA *****\n");
    AFNDProcesaEntrada(stdout, p_afnd_1);
    fprintf(stdout, "\n*****\n");

/*****
*/

    AFNDElimina(p_afnd_1);

/*****
/

    return 0;
}

```

Fíjate que el autómata tiene dos ramas, una para cada “corchete”

1. Que termina en el estado q4 para el fragmento [1*00(00)*]

- Esta rama se encarga de las cadenas que empiezan por una secuencia de cualquier cantidad (incluida 0) de 1s, terminan por un número par de ceros y no hay ningún otro símbolo entre medias.
- 2. Que termina en el estado q7 para el fragmento $[11(0+1)^*]$
 - Ésta, se encarga de las que comienzan por 11 y terminan por cualquier cadena binaria (es decir, con los dígitos 0 y 1) de cualquier longitud incluida 0.

Observa que, además de crear el autómata, se analizan las siguientes cadenas

- 111100
 - Que es reconocida por ambas ramas ya que empieza por 11 y termina por un número par de ceros.
- 1000000
 - Que sólo es reconocida por la primera, ya que no comienza por 11 pero sí termina en un número par (6) de ceros.
- 11000
 - Que sólo es reconocida por la segunda, ya que comienza por 11 y no termina en un número par de ceros.
- 01010101
 - Que no es reconocida por el autómata ya que ni empieza por 11 ni lo hace por 1 y termina a la vez en un número par de ceros.

Al ejecutar este programa la salida que se muestra es la siguiente (observa que lo añadido respecto a la práctica anterior está resaltado)

```
afll={
    num_simbolos = 2

    A={ 0 1 }

    num_estados = 8

    Q={->q0 q1 q2 q3 q4* q5 q6 q7* }

    RL++*={
        [0]    [1]    [2]    [3]    [4]    [5]    [6]    [7]
        [0]    1      1      1      0      0      1      0      0
        [1]    0      1      1      0      0      0      0      0
        [2]    0      0      1      0      0      0      0      0
        [3]    0      0      0      1      0      0      0      0
        [4]    0      0      1      0      1      0      0      0
        [5]    0      0      0      0      0      1      0      0
        [6]    0      0      0      0      0      0      1      0
        [7]    0      0      0      0      0      0      0      1
    }

    Funcion de Transición = {
        f(q0,0)={ }
        f(q0,1)={ }
        f(q1,0)={ }
        f(q1,1)={ q1 }
```

```

        f(q2,0)={ q3 }
        f(q2,1)={ }
        f(q3,0)={ q4 }
        f(q3,1)={ }
        f(q4,0)={ }
        f(q4,1)={ }
        f(q5,0)={ }
        f(q5,1)={ q6 }
        f(q6,0)={ }
        f(q6,1)={ q7 }
        f(q7,0)={ q7 }
        f(q7,1)={ q7 }
    }
}

***** PROCESA CADENA *****
[(6) 1 1 1 1 0 0]

*****

ACTUALMENTE EN {->q0 q1 q2 q5 }
[(6) 1 1 1 1 0 0]

ACTUALMENTE EN {q1 q2 q6 }
[(5) 1 1 1 0 0]

ACTUALMENTE EN {q1 q2 q7* }
[(4) 1 1 0 0]

ACTUALMENTE EN {q1 q2 q7* }
[(3) 1 0 0]

ACTUALMENTE EN {q1 q2 q7* }
[(2) 0 0]

ACTUALMENTE EN {q3 q7* }
[(1) 0]

ACTUALMENTE EN {q2 q4* q7* }
[(0)]

***** PROCESA CADENA *****

ACTUALMENTE EN {->q0 q1 q2 q5 }
[(7) 1 0 0 0 0 0 0]

ACTUALMENTE EN {q1 q2 q6 }
[(6) 0 0 0 0 0 0]

ACTUALMENTE EN {q3 }
[(5) 0 0 0 0 0]

ACTUALMENTE EN {q2 q4* }
[(4) 0 0 0 0]

ACTUALMENTE EN {q3 }
[(3) 0 0 0]

ACTUALMENTE EN {q2 q4* }
[(2) 0 0]

ACTUALMENTE EN {q3 }
[(1) 0]

ACTUALMENTE EN {q2 q4* }

```

```

[ (0) ]

*****

***** PROCESA CADENA *****

ACTUALMENTE EN {->q0 q1 q2 q5 }
[ (5) 1 1 0 0 0 ]

ACTUALMENTE EN {q1 q2 q6 }
[ (4) 1 0 0 0 ]

ACTUALMENTE EN {q1 q2 q7* }
[ (3) 0 0 0 ]

ACTUALMENTE EN {q3 q7* }
[ (2) 0 0 ]

ACTUALMENTE EN {q2 q4* q7* }
[ (1) 0 ]

ACTUALMENTE EN {q3 q7* }
[ (0) ]

*****

***** PROCESA CADENA *****

ACTUALMENTE EN {->q0 q1 q2 q5 }
[ (8) 0 1 0 1 0 1 0 1 ]

ACTUALMENTE EN {q3 }
[ (7) 1 0 1 0 1 0 1 ]

ACTUALMENTE EN {}
[ (6) 0 1 0 1 0 1 ]

*****

```

Observa que la manera de mostrar las transiciones lambda es mediante una matriz binaria que indica si desde un estado es o no accesible otro mediante transiciones lambda, tu profesor te explicará en el laboratorio cómo se hace esto.

Debes de modificar las funciones de librería que ya has codificado fundamentalmente para

- **En la parte “estática” que define el autómata**
 - Almacenar la matriz binaria de las relaciones de accesibilidad mediante lambda inducida por las transiciones lambda.
- **En la parte “dinámica” que describe el momento de análisis en el que se encuentra el autómata**
 - ...y, por lo tanto, en aquellas funciones que permitan realizar el análisis de cadenas (transitar por el siguiente símbolo, procesar la cadena, etc...)
 - Añadir al procesar cada símbolo el efecto de las transiciones lambda.

A continuación se describe con más detalle las nuevas funciones de la librería **QUE DEBES IMPLEMENTAR DE MANERA OBLIGATORIA**

```
AFND * AFNDInsertaLTransicion(  
    AFND * p_afnd,  
    char * nombre_estado_i,  
    char * nombre_estado_f );
```

- Inserta en el AFND una transición λ entre los dos estados cuyo nombre se proporciona.
- Se devuelve un puntero al AFND modificado (que se pasa como primer argumento)

```
AFND * AFNDCierraLTransicion (AFND * p_afnd);
```

- Realiza el cierre reflexivo y transitivo de la relación de accesibilidad inducida por las transiciones lambda almacenada en el AFND que se proporciona como argumento.
- Se devuelve un puntero al AFND modificado.

```
AFND * AFNDInicializaCadenaActual (AFND * p_afnd );
```

- Esta función, tal vez con otro nombre o incluida en otra, debería haberse descrito en la práctica 1
- En aquella práctica esta funcionalidad se podría haber incluido implícitamente en alguna de las otras funciones (las de inicializar el estado actual del autómata antes del procesado de una cadena, o en la misma función de procesado de la cadena).
- Su objetivo es eliminar la cadena actual dejándola vacía de forma que el autómata está preparado para especificar la siguiente cadena que se va a analizar mediante reiteradas llamadas a la función `AFNDInsertaLetra`

SE RECOMIENDA, AUNQUE NO ES IMPRESCINDIBLE que **definas** de manera explícita un **módulo (tipo abstracto de datos)** para **representar relaciones** mediante matrices cuadradas binarias y que contenga las operaciones de

- Cierre reflexivo
- Cierre transitivo

Se pide

- Cada grupo debe realizar una entrega
- El tiempo del que dispones para hacer la práctica es el correspondiente a las semanas de las próximas dos sesiones, incluida esta en la que se te presenta el enunciado.
- Tu profesor te indicará la tarea Moodle donde debes hacer la entrega.
- Cada grupo debe diseñar tantos módulos auxiliares (ficheros .c y .h) como necesite para completar la funcionalidad de la práctica, modificando para ello los que entregó al terminar la primera.
- Es imprescindible que mantengas los nombres especificados en este enunciado **sólo para las funciones que se piden de manera obligatoria**.
- Respecto a los módulos o funciones sugeridos tienes libertad de implementar estos u otros.