Grupo Jueves 15:00-17:00 semanas A

-Práctica 4-

Autor: Pablo Ernesto Angusto Delgado

NIP:842255

Autor: Miguel Aréjula Aísa

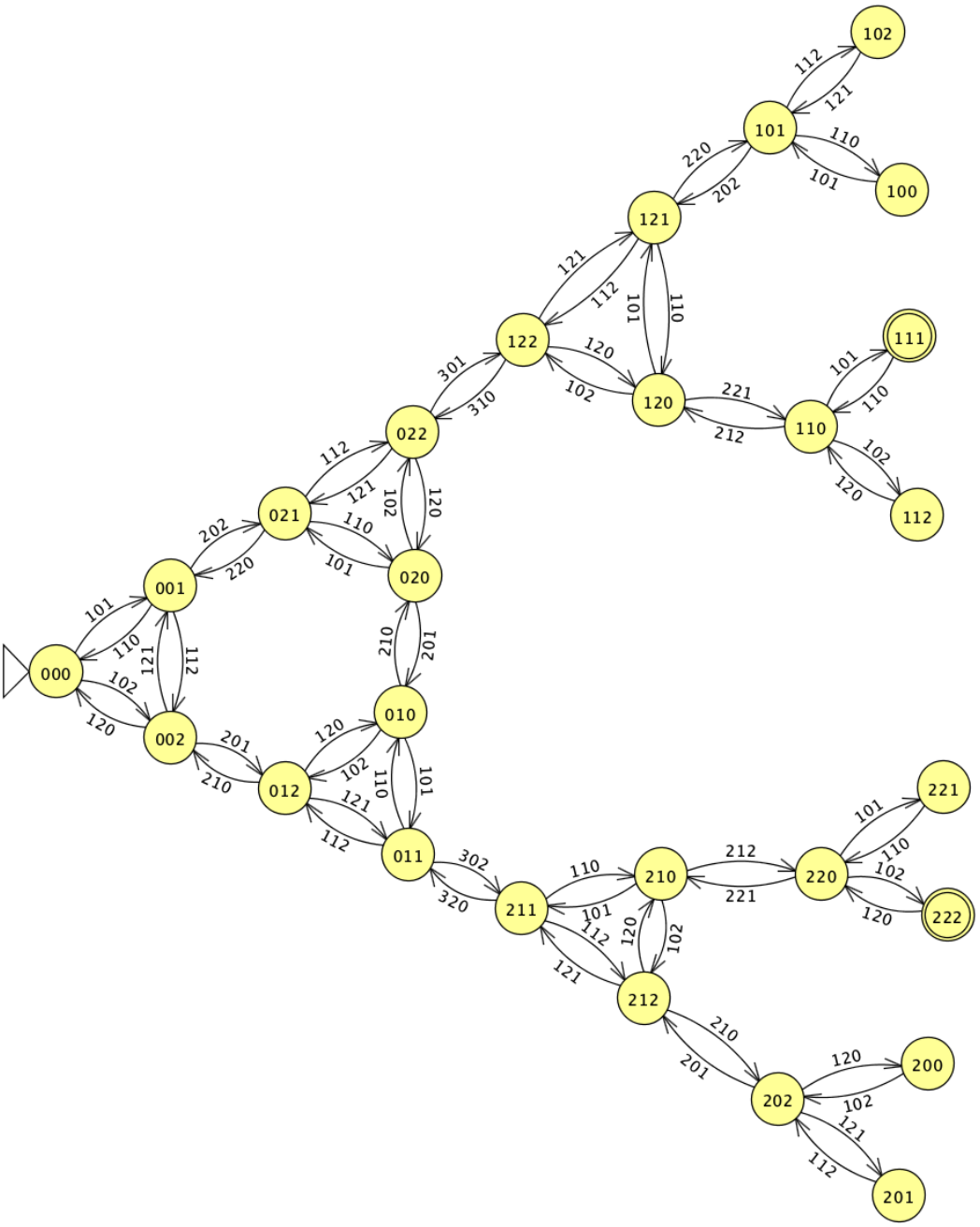
NIP: 850068

Ejercicio 1:

En este primer ejercicio se pide la especificación del problema de las Torres de Hanoi. Para ello, el primer paso que realizamos fue elegir la notación a usar. Decidimos seguir utilizando la usada en el enunciado, la cual consiste en un número de tres cifras. Cada una representa uno de los discos y su valor indica en el palo que se encuentra colocado.
El segundo paso fue plantear la lista de todos los posibles estados que se pueden alcanzar. Una vez que los posibles estados ya están definidos, escribimos todas las posibles transiciones entre los estados.

000 ->001(101), 002(102);
001 ->000(110), 002(112), 021(202);
002 ->000(120), 001(121), 012(201);
010 ->011(101), 012(102), 020(212);
011 ->010(110), 012(112), 211(302);
012 ->010(120), 011(121), 002(210);
020 ->021(101), 022(102), 010(221);
021 ->020(110), 022(112), 001(220);
022 ->020(120), 021(121), 122(301);
100 ->101(101), 102(102), 200(312);
101 ->100(110), 102(112), 121(202);
102 ->100(120), 101(121), 112(201);
110 ->111(101), 112(102), 120(212);
111 ->110(110), 112(112);
112 ->110(120), 111(121), 102(210);
120 ->121(101), 122(102), 110(221);
121 ->120(110), 122(112), 101(220);
122 ->120(120), 121(121), 022(310);
200 ->201(101), 202(102), 100(321);
201 ->200(110), 202(112), 221(202);
202 ->200(120), 201(121), 212(201);
210 ->211(101), 212(102), 220(212);
211 ->210(110), 212(112), 011(320);
212 ->210(120), 211(121), 202(210);
220 ->221(101), 222(102), 210(221);
221 ->220(110), 222(112), 201(220);
222 ->220(120), 221(121);

Previamente a diseñar el autómata final definimos el estado inicial (el estado 000) y los estados finales (los estados 111 y el 222).
Por último, realizamos el autómata del problema. Para ello hemos utilizado el programa JFLAP.


000
001
002
010
011
012
020
021
022
100
101
102
110
111
112
120
121
122
200
201
202
210
211
212
220
221
222

Ejercicio 2 :
Para realizar el ejercicio 2 hemos realizado un lenguaje que describa el grafo del autómata y está almacenado en el fichero th.l. El lenguaje se detalla a continuación:

```
%{
#include "y.tab.h"
#include <stdio.h>
%}
%%
"{"    {return(OB);}
"}"    {return(CB);}
"("    {return(OP);}
")"    {return(CP);}
";" {return(PYC);}
"->" {return(FL);}
"," {return(C);}
[0-9]+    {yylval.nombre = strdup(yytext); return(NUMBER);}
[a-z]+  {return(STRING);}
\n    {return(EOL);}
[ \t]    {/* ignorar espacios */}
.


%%
```

Por lo que cada vez que flex se encuentre un carácter contenido en este lenguaje devolverá su respectivo token. Los “{ }” sirven para dar el comienzo o el fin del grafo. Los “( )” contienen en su interior la transición correspondiente. El “;” representa el fin de los estados destino y transiciones. La “->” separa el estado actual con los próximos estados a los que podemos ir. La “,” separa los distintos estados destino. [0-9]+ representa cualquier número de uno o más cifras. [a-z]+ representa cualquiera palabra. El “\n” representa el fin de línea y por último, “\t” y “.” para ignorar los espacios y cualquier carácter extraño.
Gracias a este lenguaje podemos analizar el fichero thP3D3.txt que mediante el lenguaje DOT contiene la descripción textual del autómata del ejercicio 1 y sigue el siguiente esquema:

ESTADO ORIGEN -> ESTADO DESTINO(TRANSICIÓN), …;

Observamos que todo estado origen tiene tres estados destino excepto si estamos ante un estado perfecto, es decir, tiene los 3 discos en el mismo palo que tiene dos estados destino.
Además hemos realizado una gramática que reconoce el lenguaje creado anteriormente y se encuentra en el fichero th.y:

```
grafo:
        | grafo linea
        | STRING STRING EOL OB EOL grafo CB
        ;
```

```
linea:    orig FL lista PYC EOL
          ;

lista:    transiciones
          |  lista C transiciones
          ;

transiciones:    NUMBER OP NUMBER CP { int i; i=atoi(fila); int j;
j=atoi($1);tablaTr[funcion(i)][funcion(j)] = $3;}
                 ;

orig: NUMBER {fila=$1;}
```

En la regla orig almacenamos el valor devuelto por NUMBER en la variable fila. Además en la regla transiciones convertimos en entero el valor de fila y el del primer NUMBER y lo guardamos en base 10 en i y j respectivamente. Para pasar estos valores de base 3 a base 10 hemos realizado la función “funcion”. Finalmente, guardamos en la matriz de tipo char tablaTr en la fila i y en la columna j el valor del segundo NUMBER.
Por lo que el programa a medida que lee y reconoce el autómata mediante la gramática rellena la matriz tablaTr con las transiciones, creando la matriz de adyacencia.

Ejercicio 3:
Una vez ya tenemos la matriz de adyacencia en la matriz tablaTr para realizar el ejercicio 3, es decir, encontrar el mínimo número de movimientos necesarios para ir de un estado regular a otro. Para solucionar este problema hemos utilizado el método de elevar la matriz de adyacencia tantas veces como sea necesario hasta que el valor de la matriz en la fila “estado inicial” y columna “estado final” sea distinto de nulo. Entonces habremos encontrado un conjunto de tantos movimientos como veces hemos elevado la matriz de adyacencia.
Finalmente para realizar las potencias de la matriz de adyacencia utilizamos tablaTr: la matriz de adyacencia, Pot: la potencia n de tablaTr y Aux: necesaria porque la función “multiplicar” no puede ser de la forma multiplicar(tablaTr, pot, pot). Por lo que realizamos el siguiente conjunto de pasos para elevar la matriz de adyacencia:

```
multiplicar(tablaTr, aux, pot);
copiar(pot, aux);
```

Finalmente cuando obtengamos el conjunto de movimientos mínimo para ir de un estado regular inicial a otro final, lo mostramos por pantalla.