



Instituto Politécnico Nacional

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRÁCTICA 3: PROTOCOLO AUTOMATA

Teoría de la Computación

Autor:
Hernández Vergara Eduardo

Marzo 2022

Índice

1. Introducción	2
1.1. Objetivo	2
1.2. Automatas finitos (AFD)	2
1.3. Protocolo	2
2. Solución	3
2.1. Generacion de cadenas	3
2.2. Grafo	3
3. Código	5
3.1. Escritura de Datos	5
3.2. Grafica/Lectura Datos	7

1. Introducción

1.1. Objetivo

En la práctica desarrollada tuvimos que diseñar un protocolo, para posteriormente al protocolo ponerle el automata de paridad visto en clase.

1.2. Automatas finitos (AFD)

Un autómata finito (AF) o máquina de estado finito es un modelo computacional que realiza cálculos en forma automática sobre una entrada para producir una salida. Este modelo está conformado por un alfabeto, un conjunto de estados finito, una función de transición, un estado inicial y un conjunto de estados finales. Su funcionamiento se basa en una función de transición, que recibe a partir de un estado inicial una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un estado final o de aceptación, que representa la salida. La finalidad de los autómatas finitos es la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

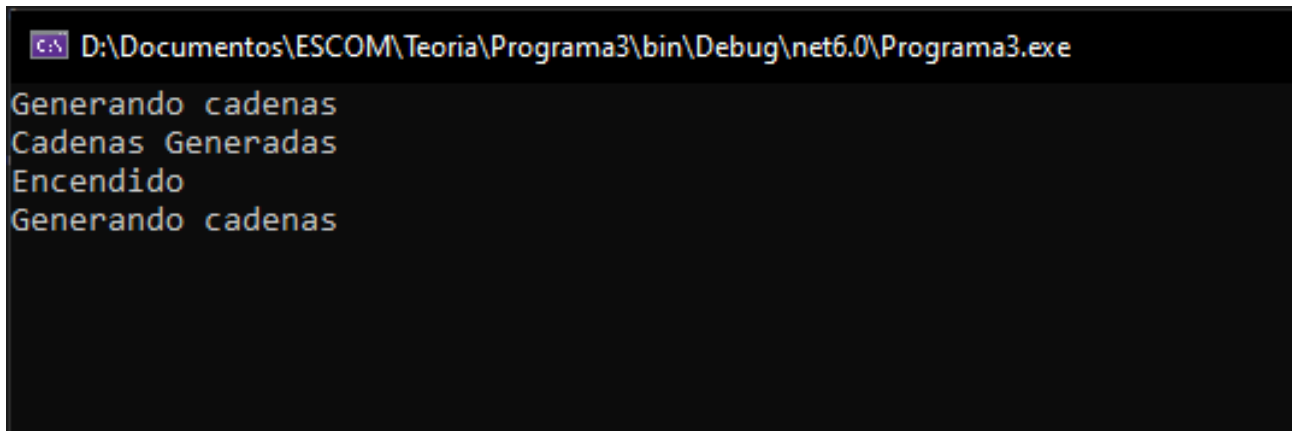
1.3. Protocolo

Los protocolos de este tipo pueden representarse mediante un autómata finito. Cada estado representa una situación en la que puede encontrarse uno de los participantes. Es decir, el estado recuerda qué sucesos importantes han ocurrido y qué sucesos todavía no han tenido lugar. Las transiciones entre estados se producen cuando tiene lugar uno de los cinco sucesos descritos anteriormente. Supondremos que estos sucesos son externos al autómata que representa a los tres participantes, aunque cada participante sea responsable de iniciar uno o más de los sucesos. Lo importante en este problema es qué secuencias pueden ocurrir y no quién las inicia.

2. Solución

Para la implementación de la solución se uso C#, con .Net 6.0, con el IDE Visual Studio 2022, como veremos a continuación con las capturas de resultado, veremos como salio.

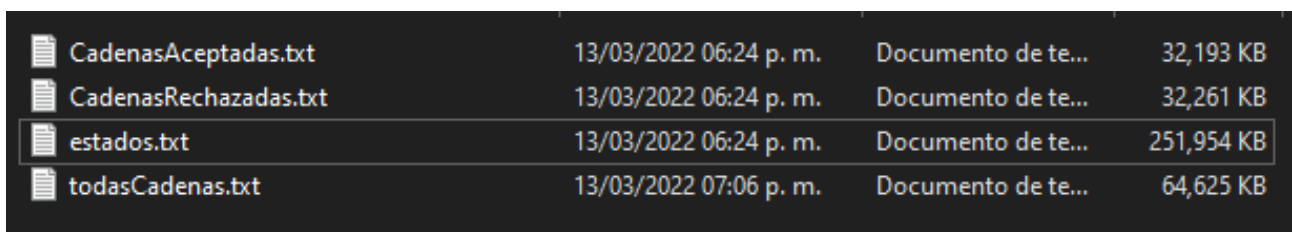
2.1. Generacion de cadenas



```
C:\> D:\Documentos\ESCOM\Teoria\Programa3\bin\Debug\net6.0\Programa3.exe
Generando cadenas
Cadenas Generadas
Encendido
Generando cadenas
```

Figura 1: Solos nos muestra que se generaron

En esta captura se nos muestra que en la consola solo nos aparece el texto de si se generan o no, ademas tambien nos dice si ya se generaron las cadenas y si se apaga o enciende la máquina. En esta captura se ven los tamaños de los archivos.



CadenasAceptadas.txt	13/03/2022 06:24 p. m.	Documento de te...	32,193 KB
CadenasRechazadas.txt	13/03/2022 06:24 p. m.	Documento de te...	32,261 KB
estados.txt	13/03/2022 06:24 p. m.	Documento de te...	251,954 KB
todasCadenas.txt	13/03/2022 07:06 p. m.	Documento de te...	64,625 KB

Figura 2: El tamaño del archivo

2.2. Grafo

En la siguiente captura, podemos ver lo que seria el grafo:

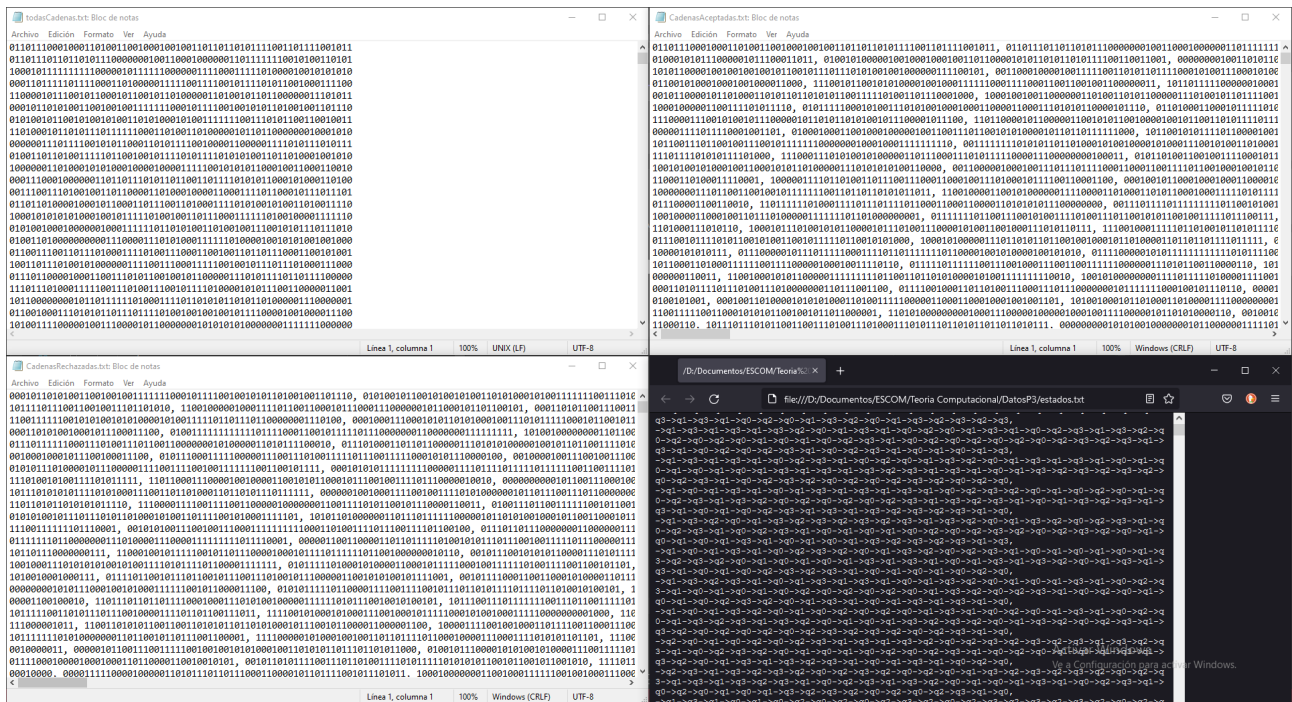


Figura 3: Aquí podemos ver los archivos de cadenas

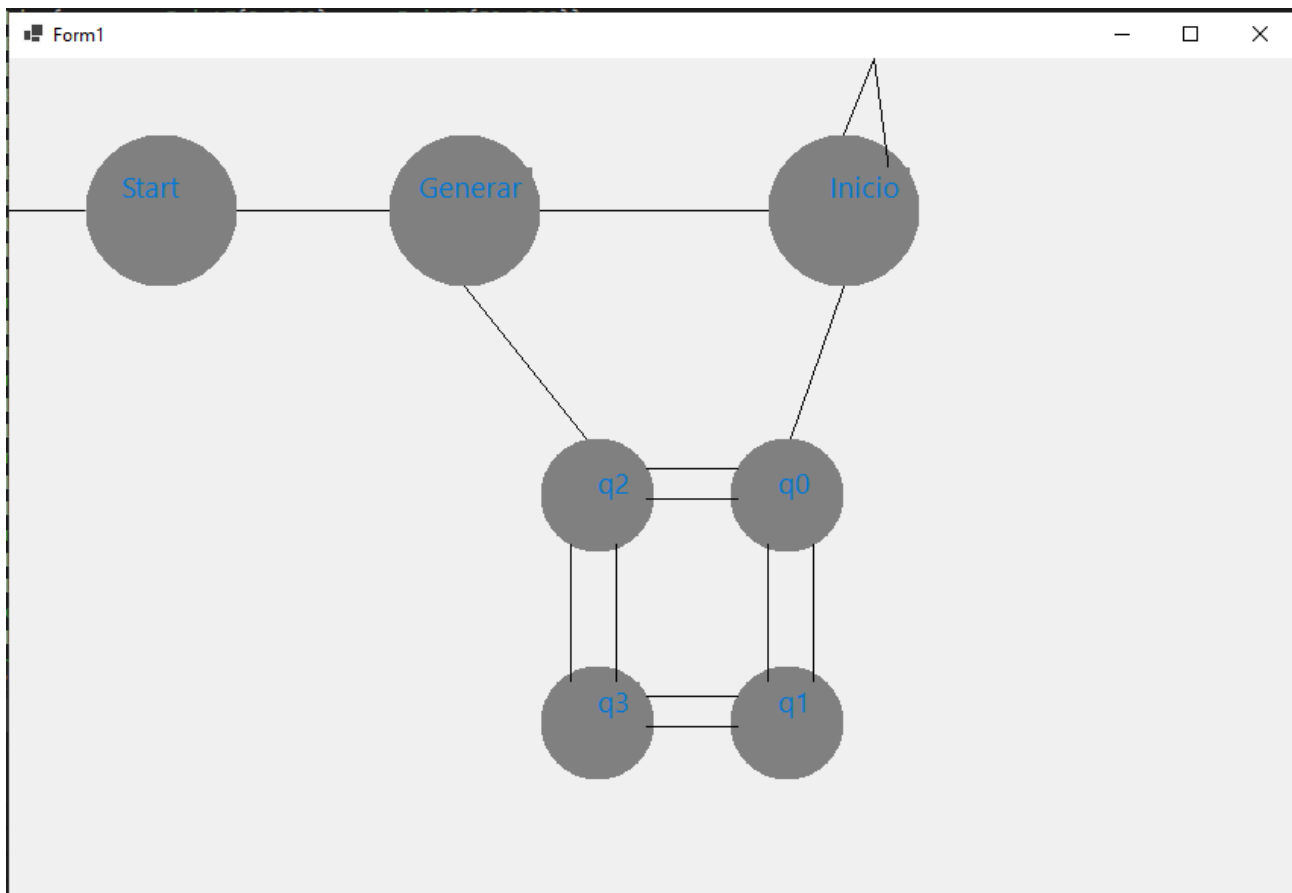


Figura 4: Grafo

3. Código

3.1. Escritura de Datos

Listing 1: Escritura de Datos C# .net 6.0

```
1 using System;
2 using System.Linq;
3
4 namespace Programa3
5 {
6     class Programa3
7     {
8         static void Main(string[] args)
9         {
10             GenerarCadenas();
11         }
12         static void Iniciar(){
13             var rand = new Random();
14             if (rand.Next(2) == 1){
15                 Console.WriteLine("Apagado");
16             }
17             else {
18                 Console.WriteLine("Encendido");
19                 GenerarCadenas();
20             }
21         }
22     }
23     static void GenerarCadenas() {
24         var characters = "01";
25         var Charsarr = new char[64];
26         var random = new Random();
27         Console.WriteLine("Generando cadenas");
28         for (int j = 0; j < 1000000; j++){
29             for (int i = 0; i < Charsarr.Length; i++){
30                 Charsarr[i] = characters[random.Next(characters.Length
31                     )];
32             }
33             var resultString = new String(Charsarr);
34             File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
35                 Computacional\\DatosP3\\todasCadenas.txt",
36                 resultString + "\n");
37         }
38         Console.WriteLine("Cadenas Generadas");
39         Console.Beep(37, 1000);
40         automata();
41     }
42     static void automata()
43     {
44         StreamReader lectura = new StreamReader("D:\\Documentos\\ESCOM
45             \\Teoria Computacional\\DatosP3\\todasCadenas.txt");
46         string linea;
47         while ((linea = lectura.ReadLine()) != null)
48         {
49             string auto = "q0", estado = "";
50             for (int c = 0; c < linea.Length; c++)
51             {
52                 if (auto.Equals("q0") == true)
```

```

50     {
51         if (linea[c].Equals('0') == true)
52         {
53             auto = "q2";
54             estado = estado + "->" + auto;
55         }
56         else if (linea[c].Equals('1') == true)
57         {
58             auto = "q1";
59             estado = estado + "->" + auto;
60         }
61     }
62     else if (auto.Equals("q1") == true)
63     {
64         if (linea[c].Equals('0') == true)
65         {
66             auto = "q3";
67             estado = estado + "->" + auto;
68         }
69         else if (linea[c].Equals('1') == true)
70         {
71             auto = "q0";
72             estado = estado + "->" + auto;
73         }
74     }
75     else if (auto.Equals("q2") == true)
76     {
77         if (linea[c].Equals('0') == true)
78         {
79             auto = "q0";
80             estado = estado + "->" + auto;
81         }
82         else if (linea[c].Equals('1') == true)
83         {
84             auto = "q3";
85             estado = estado + "->" + auto;
86         }
87     }
88     else if (auto.Equals("q3") == true)
89     {
90         if (linea[c].Equals('0') == true)
91         {
92             auto = "q1";
93             estado = estado + "->" + auto;
94         }
95         else if (linea[c].Equals('1') == true)
96         {
97             auto = "q2";
98             estado = estado + "->" + auto;
99         }
100     }
101 }
102 }
103 if (auto.Equals("q0") == true)
104 {
105     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
        Computacional\\DatosP3\\CadenasAceptadas.txt",
        linea + ", ");
106     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
        Computacional\\DatosP3\\estados.txt", estado + ",

```

```

107         });
108     else
109     {
110         File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
            Computacional\\DatosP3\\CadenasRechazadas.txt",
            linea + ", ");
111         File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
            Computacional\\DatosP3\\estados.txt", estado + ",
            ");
112     }
113 }
114
115 }
116 Iniciar();
117 }
118
119 }
120 }

```

3.2. Grafica/Lectura Datos

Listing 2: Grafo C# (.net 6.0)

```

1 namespace Graph
2 {
3     public partial class Form1 : Form
4     {
5         public Form1()
6         {
7             InitializeComponent();
8         }
9
10        private void Form1_Paint(object sender, PaintEventArgs e)
11        {
12            Graphics g = e.Graphics;
13            Pen pen = new Pen(Color.Black);
14            Brush brush = new SolidBrush(Color.Gray);
15            g.DrawLine(pen, new PointF(0, 100), new PointF(50, 100));
16            g.FillEllipse(brush, 50, 50, 100, 100);
17            g.FillEllipse(brush, 250, 50, 100, 100);
18            g.FillEllipse(brush, 500, 50, 100, 100);
19            g.FillEllipse(brush, 475, 250, 75, 75);
20            g.FillEllipse(brush, 475, 400, 75, 75);
21            g.FillEllipse(brush, 350, 250, 75, 75);
22            g.FillEllipse(brush, 350, 400, 75, 75);
23            g.DrawLine(pen, new PointF(150, 100), new PointF(250, 100));
24            g.DrawLine(pen, new PointF(350, 100), new PointF(500, 100));
25            g.DrawLine(pen, new PointF(550, 150), new PointF(515, 250));
26            //automata
27            g.DrawLine(pen, new PointF(530, 320), new PointF(530, 410));
28            g.DrawLine(pen, new PointF(500, 320), new PointF(500, 410));
29            //automata
30            g.DrawLine(pen, new PointF(370, 320), new PointF(370, 410));
31            g.DrawLine(pen, new PointF(400, 320), new PointF(400, 410));
32            //automata
33            g.DrawLine(pen, new PointF(420, 270), new PointF(480, 270));
34            g.DrawLine(pen, new PointF(420, 290), new PointF(480, 290));

```



```
35         //automata
36         g.DrawLine(pen, new PointF(420, 420), new PointF(480, 420));
37         g.DrawLine(pen, new PointF(420, 440), new PointF(480, 440));
38         //protocolo
39         g.DrawLine(pen, new PointF(380, 250), new PointF(300, 150));
40         //esperar
41         g.DrawLine(pen, new PointF(550, 50), new PointF(570, 0));
42         g.DrawLine(pen, new PointF(570, 0), new PointF(580, 75));
43     }
44
45     private void label1_Click(object sender, EventArgs e)
46     {
47
48     }
49 }
50 }
```
