
Teoría computacional

- Prácticas tercer parcial -

Reportes prácticas del tercer parcial
2CM5/Vargas Romero Erick Efraín
Prof. Juárez Martínez Genaro

Instituto Politécnico Nacional
Escuela Superior de Cómputo
Juan de Dios Bátiz, nueva industrial Vallejo
07738 ciudad de México

0.1 Gramáticas no ambiguas

Una gramática no ambigua puede definirse o bien es una gramática libre de contexto para la que cada cadena válida posee una derivación a la izquierda. Muchos lenguajes admiten tanto gramáticas ambiguas como no ambiguas mientras que otros lenguajes admiten solo gramáticas ambiguas. Cualquier lenguaje que no es vacío admite una gramática ambigua al tomar una gramática no ambigua e introducir una regla de derivación con duplicidad.

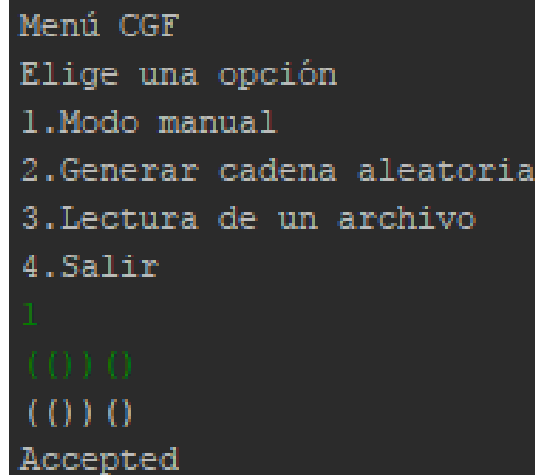
0.1.1 Balanceo de paréntesis

Este programa hace uso de una gramática no ambigua, la cual se ha definido en la sección anterior. Siendo así es necesario mencionar que este programa contiene un pequeño menú. Este menú tiene cuatro opciones: modo manual, generar una cadena aleatoria, lectura de un archivo y salir.

A continuación se muestra el funcionamiento del programa, primeramente utilizando la primer opción e nuestro menú, el cual es modo manual.

En la figura uno se puede mostrar el menú que previamente se ah mencionado, el cual contiene cuatro opciones, de las cuales las primeras tres son de gran relevancia.

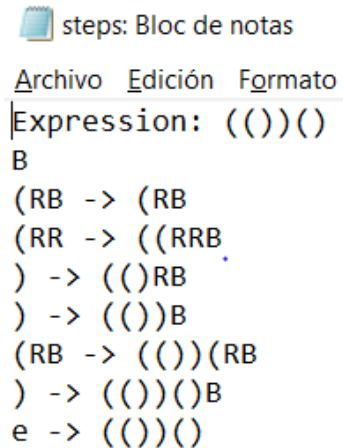
En la misma figura podemos observar que se ha seleccionado la opción uno, además de que se ha ingresado la cadena `(())()` para realizar una prueba del funcionamiento del programa, además podemos observar que es una cadena válida y en consola se ve reflejado el que sea válida. Este proceso es realizado únicamente evaluando cada carácter que contiene nuestra cadena, en simples palabras se tienen ifs anidados para evaluar el carácter que ha sido ingresado y para realizar la sustitución del símbolo que ha sido evaluado. Las reglas de derivación utilizadas han sido establecidas como constantes en arreglos de Strings para hacer más fácil el uso de estas.



```
Menú CGF
Elige una opción
1.Modos manual
2.Generar cadena aleatoria
3.Lectura de un archivo
4.Salir
1
(( ))()
Accepted
```

Figure 1: Modo manual

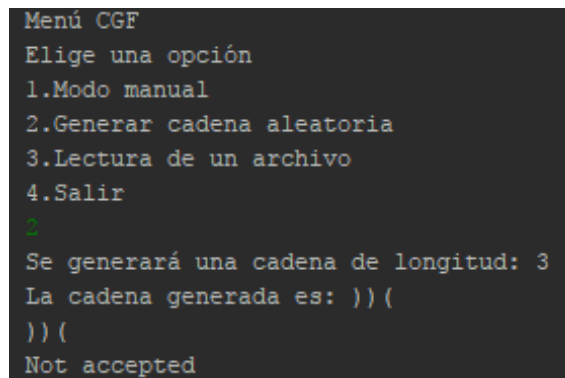
En la siguiente figura, podemos observar todo el procedimiento que se ha seguido para la verificación de la cadena. Esto podemos verlo en un archivo de texto llamado steps. En el archivo se muestra la regla que se ha utilizado antes de la flecha, y después de esta se muestra el procedimiento.



```
steps: Bloc de notas
Archivo Edición Formato
Expression: (())(()B
(RB -> (RB
(RR -> ((RRB
) -> (()RB
) -> (()B
(RB -> (())(RB
) -> (())(B
e -> (())(
```

Figure 2: Prueba en manual

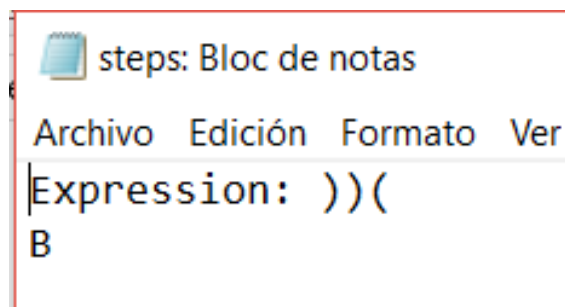
Ahora se procede a utilizar la opción dos, la cual genera una cadena aleatoria formada únicamente por paréntesis, la cual tiene como tamaño máximo una dimensión de 1000, pero para esta prueba se ha reducido a 10 para que se pueda apreciar este modo automático.



```
Menú CGF
Elige una opción
1.Modos manual
2.Generar cadena aleatoria
3.Lectura de un archivo
4.Salir
2
Se generará una cadena de longitud: 3
La cadena generada es: ))(
)) (
Not accepted
```

Figure 3: Modo automático

Al igual que en el modo manual se muestra a continuación el contenido del fichero steps



```
steps: Bloc de notas
Archivo Edición Formato Ver
Expression: ))(
B
```

Figure 4: Contenido steps.txt

Finalmente hacemos uso del modo "Lectura de un archivo", primeramente se muestra el contenido del fichero que será leído por nuestro programa, el cual contiene diferentes cadenas, algunas válidas otras no. Esto con el fin de intentar que nuestro programa entre en estado de error y finalmente sea finalizado.

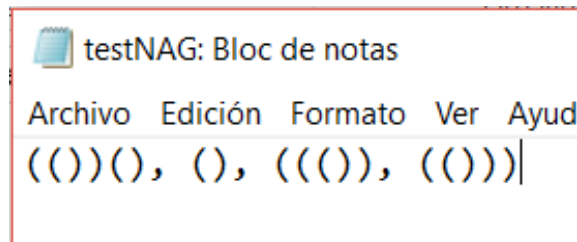


Figure 5: Contenido testNAG.txt

A continuación se muestra como es seleccionada la opción deseada. Y en consola se nos muestra si la cadena fue o no aceptada. Y como podemos observar, se han detectado las cuatro cadenas que estan contenidas en el fichero steps.txt, siendo así la primer cadena válida, la segunda también, la tercera y cuarta inválidas.

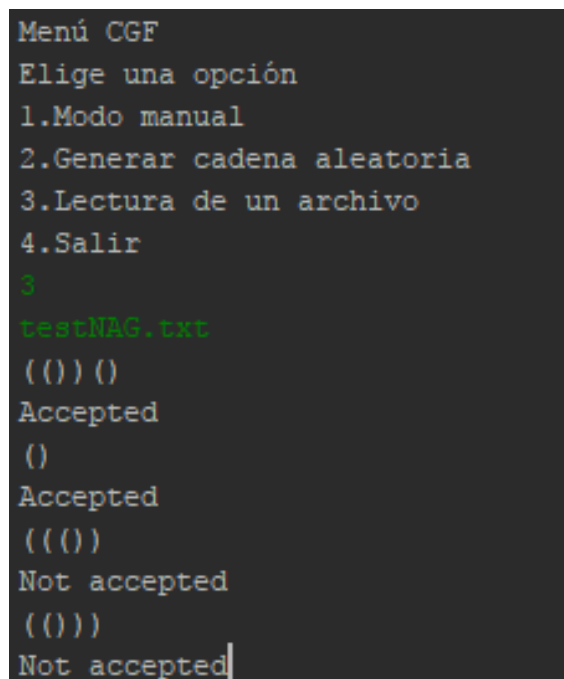


Figure 6: Lectura de un archivo

Al igual que las dos pruebas anteriores se muestra el contenido del fichero steps.txt

```

steps: Bloc de notas
Archivo Edición Formato
Expression: (()>()
B
(RB -> (RB
(RR -> ((RRB
) -> (()RB
) -> (()B
(RB -> (())(RB
) -> (()>()B
e -> (()>()
Accepted

Expression: ()
B
(RB -> (RB
) -> ()B
e -> ()
Accepted

Expression: ((())
B
(RB -> (RB
(RR -> ((RRB
(RR -> (((RRRB
) -> ((()RRB
) -> ((())RB
Expression: ((())
B
(RB -> (RB
(RR -> ((RRB
) -> (()RB
) -> (()B

```

Figure 7: Contenido steps.txt

A continuación se muestra el código utilizado en este programa. Empezamos por la clase CGF, la cual contiene nuestro menú y el main de este programa.

```

1 package cgf;
2
3 import java.util.Scanner;
4
5 public class CGF {
6
7     public static final int constant = 10;
8

```

```

9      public static void main(String[] args) {
10         boolean flag = true;
11         while(flag){
12             Scanner scanner = new Scanner(System.in);
13             System.out.println("¿Men CGF");
14             System.out.println("Elige una opción");
15             System.out.println("1.Modos manual");
16             System.out.println("2.Generar cadena aleatoria");
17             System.out.println("3.Lectura de un archivo");
18             System.out.println("4.Salir");
19             switch(scanner.nextInt()){
20                 case 1:
21                     scanner.nextLine();
22                     new expression().evaluateExpression(scanner.nextLine(),
23                                                         0);
24                     break;
25                 case 2:
26                     new expression().evaluateExpression(randomString(), 0);
27                     break;
28                 case 3:
29                     scanner.nextLine();
30                     new expression().readByFile(scanner.nextLine());
31                     break;
32                 case 4:
33                     flag = false;
34                     break;
35                 default:
36                     System.out.println("Esa opción no existe");
37                     break;
38             }
39         }
40     }
41
42     private static String randomString(){
43         int random = (int)(Math.random()*constant);
44         String str = "";
45         System.out.println("Se generará una cadena de longitud: " + random);
46         for(int i = 0; i < random; i++){
47             if(Math.random() < 0.5){
48                 str += "(";
49             } else {
50                 str += ")";
51             }
52         }
53         System.out.println("La cadena generada es: " + str);
54         return str;
55     }
56 }

```

La siguiente clase es la que hace la "magia" de este programa. Ya que es donde se evalúa la expresión.

```

1 package cgf;
2
3 import Palindrome.forFiles.MyFiles;
4 import java.io.FileReader;
5
6 public class expression {
7
8     private MyFiles file = new MyFiles("Parse Trees/steps.txt");

```

```

9
10 String DRfB[] = {"(RB", "e"}, DRfR[] = {")", "(RR"};
11
12 public void evaluateExpression(String strParentheses, int mode) {
13     strParentheses = strParentheses.trim();
14     String process = "B", ruleSelected = "";
15
16     if (!strParentheses.isEmpty()) {
17         file.writeSth("Expression: " + strParentheses);
18         file.writeSth(process);
19
20         System.out.println(strParentheses);
21         strParentheses += ' ';
22         for (int i = 0; i < strParentheses.length(); i++) {
23             char character = strParentheses.charAt(i);
24             int positionSymbol = positionFirstCharacter(process);
25             if (positionSymbol >= 0) {
26                 if (character == '(') {
27
28                     if (process.charAt(positionSymbol) == 'B') {
29                         ruleSelected = DRfB[0];
30                         process = process.replaceFirst("B", ruleSelected)
31                         ;
32                     } else if (process.charAt(positionSymbol) == 'R') {
33                         ruleSelected = DRfR[1];
34                         process = process.replaceFirst("R", ruleSelected)
35                         ;
36                     } else {
37                         System.out.println("Not accepted");
38                         break;
39                     }
40                 } else if (character == ')') {
41
42                     if (process.charAt(positionSymbol) == 'R') {
43                         ruleSelected = DRfR[0];
44                         process = process.replaceFirst("R", ruleSelected)
45                         ;
46                     } else {
47                         System.out.println("Not accepted");
48                         break;
49                     }
50                 }
51             } else if (character == 32) {
52                 if (process.charAt(positionSymbol) == 'B') {
53                     ruleSelected = DRfB[1];
54                     process = process.replaceFirst("B", " ");
55                     System.out.println("Accepted");
56                 } else {
57                     System.out.println("Not accepted");
58                     break;
59                 }
60             } else {
61                 System.out.println("Not accepted, invalid character")
62                 ;
63             }
64             file.writeSth(ruleSelected + " -> " + process);
65             if (ruleSelected.equals(DRfB[1]))
66                 file.writeSth("Accepted \r\n");
67         }
68     }
69 }

```



```

66         }
67         if (mode == 0) {
68             file.closeFile();
69         }
70     }
71 }
72
73 public int positionFirstCharacter(String process) {
74     for (int i = 0; i < process.length(); i++) {
75         if (process.charAt(i) == 'R' || process.charAt(i) == 'B') {
76             return i;
77         }
78     }
79     return -1;
80 }
81
82 public void readByFile(String path) {
83     String str = "";
84     try {
85         FileReader fileReader = new FileReader(path);
86         int character = fileReader.read();
87
88         while (character != -1) {
89
90             if (character == '(' || character == ')') {
91                 str += (char) character;
92             } else {
93                 evaluateExpression(str, 1);
94                 str = "";
95             }
96             character = fileReader.read();
97         }
98
99         if (character == '(' || character == ')') {
100             str += (char) character;
101         } else {
102             evaluateExpression(str, 1);
103             str = "";
104         }
105
106         fileReader.close();
107     } catch (Exception e) {
108         System.err.println("Error reading file: " + path);
109         System.err.println(e);
110     }
111     file.closeFile();
112 }
113 }

```

0.2 Máquina de Turing

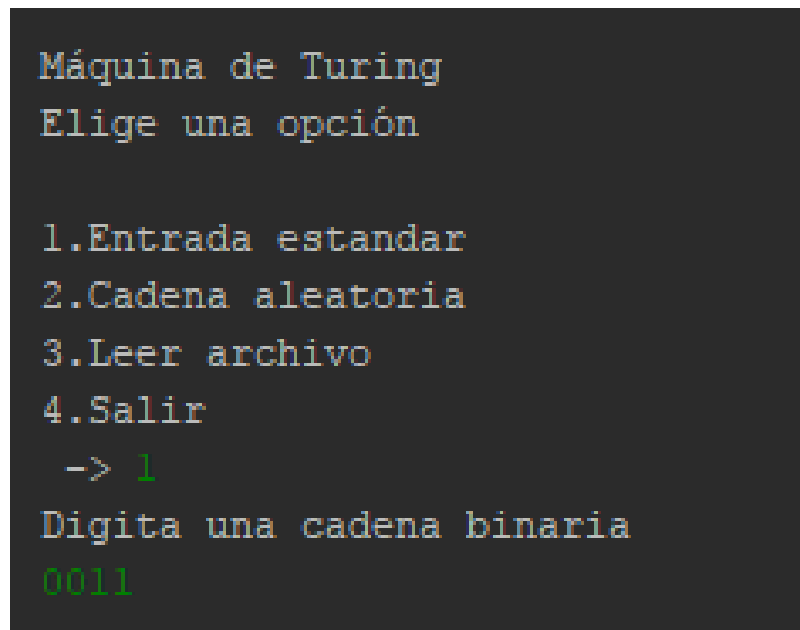
La máquina de Turing es un dispositivo que manipula símbolos sobre una cinta, esto según las reglas. A pesar de la simplicidad que posee, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de una computadora y además es bastante útil para la explicación de funciones de una CPU dentro de una computadora.

La máquina de Turing es tan poderosa que incluso es capaz de simular cualquier otra máquina de Turing, siendo así es llamada una máquina universal de Turing.

0.2.1 Máquina de turing 0s y 1s

El siguiente programa es bastante simple, ya que la máquina de Turing a construir para este programa aceptará el lenguaje $\{0^n 1^n | n \geq 1\}$. Inicialmente, se le proporciona a la cinta una secuencia finita de ceros y unos, precedida y seguida por secuencias infinitas de espacios en blanco. Alternativamente, la máquina de Turing cambiará primero un 0 por X y luego un 1 por una Y, hasta que se hayan cambiado todos los ceros y unos.

En la siguiente figura podemos apreciar que nuestro programa tiene un menú, el cual tiene cuatro opciones: entrada estándar, cadena aleatoria, leer archivo y salir. Para esta prueba se hace uso de la primera opción y además se ingresa la cadena 0011 para iniciar con las pruebas.



```

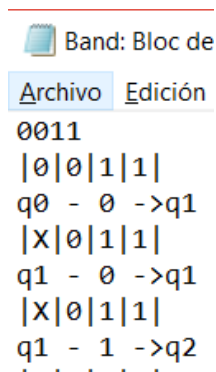
Máquina de Turing
Elige una opción

1.Entrada estandar
2.Cadena aleatoria
3.Leer archivo
4.Salir
-> 1
Digita una cadena binaria
0011

```

Figure 8: Modo manual

Al igual que los programas anteriores, se almacena en un archivo de texto todo lo que se ha realizado para poder llegar al resultado deseado, para nuestra máquina de Turing en la figura nueve, podemos apreciar lo mismo.



```

Band: Bloc de
Archivo Edición
0011
|0|0|1|1|
q0 - 0 ->q1
|X|0|1|1|
q1 - 0 ->q1
|X|0|1|1|
q1 - 1 ->q2
. . . .

```

Figure 9: Contenido Band.txt

Ahora se utilizará la opción dos, la cual genera una cadena aleatoria compuesta por ceros y unos. Para esta prueba se da una longitud máxima de cien la cual es introducida en la cinta de nuestra máquina de Turing y finalmente evaluada. Como podemos observar esta cadena no es válida y esto se ha determinado en el estado tres, siendo así el programa es finalizado por completo.

```
Máquina de Turing
Elige una opción

1.Entrada estandar
2.Cadena aleatoria
3.Leer archivo
4.Salir
-> 2
El sistema generó la cadena: 00101110011101101
q3 Cadena inválida
```

Figure 10: Modo automático

Al igual que la prueba anterior mostramos el contenido del fichero Band.txt el cual muestra la cadena que se ha evaluado y los pasos que se han seguido, para todos los casos no se mostrará el fichero por completo debido a que hay en algunos casos demasiados pasos que se han seguido para determinar la validez de la cadena.



Band: Bloc de notas

Archivo Edición Formato Ver Ayuda

|00101110011101101

|0|0|1|0|1|1|1|0|0|1|1|1|0|1|1|0|1|

q0 - 0 ->q1

|X|0|1|0|1|1|1|0|0|1|1|1|0|1|1|0|1|

q1 - 0 ->q1

|X|0|1|0|1|1|1|0|0|1|1|1|0|1|1|0|1|

q1 - 1 ->q2

|X|0|Y|0|1|1|1|0|0|1|1|1|0|1|1|0|1|

q2 - 0 ->q2

Figure 11: Contenido Band.txt

Finalmente, se hará la prueba del modo "Leer un archivo", pero primeramente se mostrará el contenido del fichero test.txt el cual tiene algunas cadenas binarias las cuales evaluará el programa.

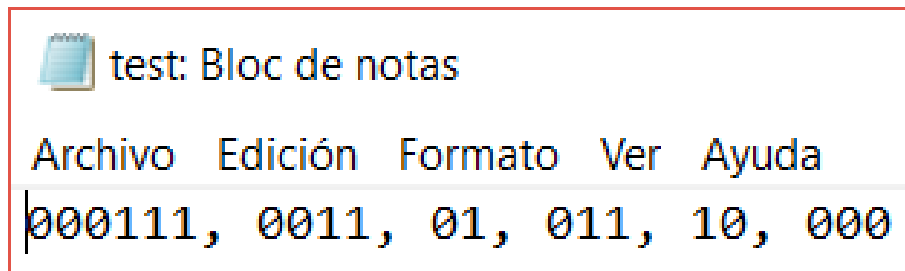


Figure 12: Contenido test.txt

Una vez hecho lo anterior procedemos a seleccionar la opción "Leer archivo" y finalmente digitamos la ruta del fichero que nuestro programa va a leer. Posteriormente el programa inicia la evaluación de las cadenas contenidas en nuestro archivo de texto. Como podemos observar al igual que en el caso anterior, se ha leído una cadena que no es válida, y a causa de esto el programa ha "muerto" en el estado q_3 .

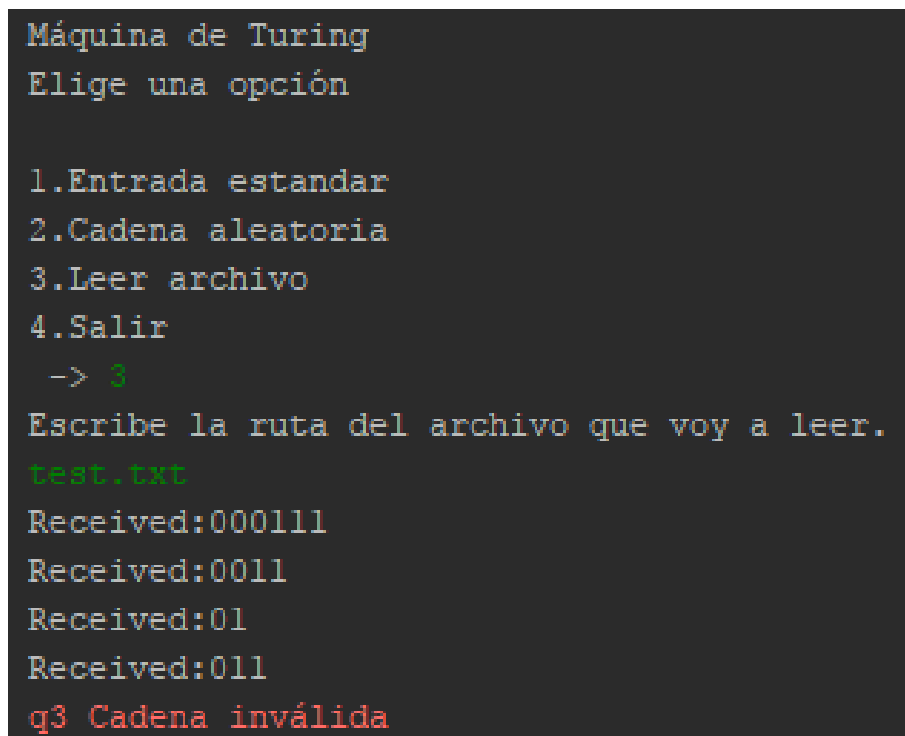


Figure 13: Prueba "Leer archivo"

Finalmente y como se ha hecho en los casos anteriores, tenemos el contenido del fichero Band.txt, el contenido es mostrado en cuatro imágenes diferentes ya que el contenido es extenso. Primer parte para la cadena 000111

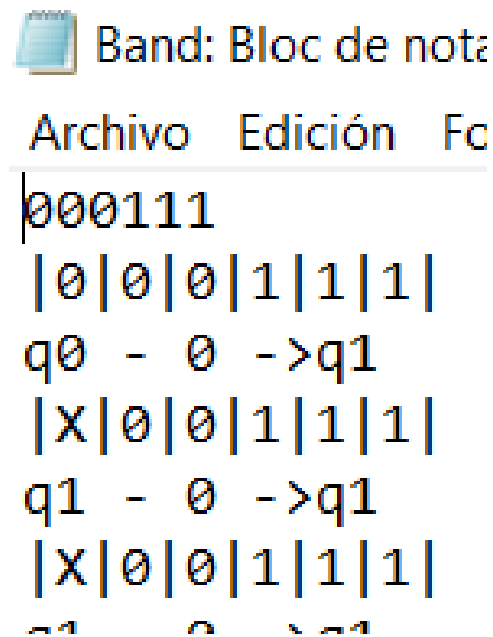


Figure 14: Contenido Band.txt parte 1

Segunda parte para la cadena 0011

```
0011
|0|0|1|1|
q0 - 0 -> q1
    |X|0|1|1|
q1 - 0 -> q1
    |X|0|1|1|
q1 - 1 -> q2
```

Figure 15: Contenido Band.txt parte 2

Tercer parte para la cadena 01

```

01
|0|1|
q0 - 0 ->q1
|X|1|
q1 - 1 ->q2
|X|Y|
q2 - X ->q0

```

Figure 16: Contenido Band.txt parte 3

Cuarta parte para la cadena 011

```

011
|0|1|1|
q0 - 0 ->q1
|X|1|1|
q1 - 1 ->q2
|X|Y|1|

```

Figure 17: Contenido Band.txt parte 4

Finalmente tenemos el código utilizado en este programa. Iniciando con la clase TuringMachine el cual tiene contenido el código del menú mostrado en las imágenes.

```

1 package turingmachine;
2
3 import java.util.Scanner;
4
5 public class TuringMachine {
6
7     private static final int constant = 100;
8     public static void main(String[] args) {
9
10         boolean flag = true;
11
12         while(flag){
13             Scanner scanner = new Scanner(System.in);
14             System.out.println("\n\nánMquina de Turing");
15             System.out.println("Elige una opción");
16             System.out.println("\n1.Entrada estandar");
17             System.out.println("2.Cadena aleatoria");
18             System.out.println("3.Leer archivo");
19             System.out.println("4.Salir");

```

```

20         System.out.print(" -> ");
21         switch(scanner.nextInt()){
22             case 1:
23                 scanner.nextLine();
24                 System.out.println("Digita una cadena binaria");
25                 new Machine().standardInput(scanner.nextLine(), 0);
26                 break;
27             case 2:
28                 scanner.nextLine();
29                 new Machine().standardInput(randomString(), 0);
30                 break;
31             case 3:
32                 scanner.nextLine();
33                 System.out.println("Escribe la ruta del archivo que voy a
34                     leer.");
35                 new Machine().readByKeyboard(scanner.nextLine());
36             case 4:
37                 System.exit(0);
38                 break;
39             default:
40                 System.out.println("Esa opción no existe");
41                 break;
42         }
43     }
44 }
45
46 public static String randomString(){
47     String str = "";
48     for(int i = 0; i < (int)(Math.random()*constant); i++){
49         str += (int)(Math rint(Math.random()));
50     }
51     System.out.println("El sistema ógener la cadena: " + str);
52     return str;
53 }

```

Finalmente tenemos la clase Machine la cual hace la "magia" de este programa.

```

1 package turingmachine;
2
3 import Files.MyFile;
4 import java.io.FileReader;
5
6 public class Machine {
7
8     private MyFile file = new MyFile("Turing Machine/Band.txt");
9     private static final String q0 = "q0", q1 = "q1", q2 = "q2", q3 = "q3",
10         q4 = "q4";
11
12     private int position = 0;
13     private char band[];
14
15     public void standardInput(String str, int mode){
16         System.out.println("Received: " + str);
17         file.writeSth(str);
18         str = str.trim();
19         str = str + ' ';
20         band = str.toCharArray();
21         String actualState = q0;
22         for(position = 0; position < str.length(); /**Param unnecessary**/){

```

```

23         actualState = states(actualState, band[position]);
24     }
25     if(actualState.equals(q4))
26         file.writeSth("Accepted");
27     else file.writeSth("Not accepted");
28
29     bandToFile();
30     if(mode == 0)
31         file.closeFile();
32 }
33
34 public void readByKeyboard(String str) {
35     String combination = "";
36     try{
37
38         FileReader fileReader = new FileReader(str);
39         int character = fileReader.read();
40
41         while(character != -1){
42             if((char)character == '0' || (char)character == '1')
43                 combination+=(char)character;
44             else if(!combination.trim().isEmpty()){
45                 standardInput(combination, 1);
46                 combination = "";
47             }
48             character = fileReader.read();
49         }
50         if((char)character == '0' || (char)character == '1')
51             combination+=(char)character;
52         else if(!combination.trim().isEmpty()){
53             standardInput(combination, 1);
54             combination = "";
55         }
56         character = fileReader.read();
57     } catch (Exception e){
58         System.err.println("Error reading file: " + str + " error " + e);
59     } finally {
60         file.closeFile();
61     }
62 }
63
64 private String states(String actualState, char symbol){
65     String newState = "";
66     switch(actualState){
67         case q0:
68             newState = q0(actualState, symbol);
69             break;
70         case q1:
71             newState = q1(actualState, symbol);
72             break;
73         case q2:
74             newState = q2(actualState, symbol);
75             break;
76         case q3:
77             newState = q3(actualState, symbol);
78             break;
79         default:
80             System.err.println("Nothing todo");
81             file.closeFile();
82             System.exit(0);
83             break;

```



```

84     }
85
86     return newState;
87 }
88
89 private String q0(String actualState, char symbol){
90     String newState = "";
91
92     //Change symbol for an X;
93     if(symbol == '0'){
94         newState = q1;
95         band[position] = 'X';
96         toRight();
97     } else if(symbol == 'Y'){
98         newState = q3;
99         band[position] = 'Y';
100        toRight();
101    } else {
102        System.err.println("q0 Cadena áinvlida");
103        file.closeFile();
104        System.exit(0);
105    }
106    file.writeSth(actualState + " - " + symbol + " ->" + newState);
107    return newState;
108 }
109
110 private String q1(String actualState, char symbol){
111     String newState = "";
112
113     if(symbol == '0'){
114         newState = q1;
115         //band[position] = '0';
116         toRight();
117     } else if(symbol == '1'){
118         newState = q2;
119         band[position] = 'Y';
120         toLeft();
121     } else if(symbol == 'Y'){
122         newState = q1;
123         toRight();
124     } else {
125         System.err.println("q1 Cadena áinvlida");
126         file.closeFile();
127         System.exit(0);
128     }
129     file.writeSth(actualState + " - " + symbol + " ->" + newState);
130     return newState;
131 }
132
133 private String q2(String actualState, char symbol){
134     String newState = "";
135
136     if(symbol == '0'){
137         newState = q2;
138         toLeft();
139     } else if(symbol == 'X'){
140         newState = q0;
141         toRight();
142     } else if(symbol == 'Y'){
143         newState = q2;
144         toLeft();

```

```

145     } else {
146         System.out.println("q2 cadena áinvlida");
147         file.closeFile();
148         System.exit(0);
149     }
150     file.writeSth(actualState + " - " + symbol + " ->" + newState);
151     return newState;
152 }
153
154 private String q3(String actualState, char symbol){
155     String newState = "";
156
157     if(symbol == 'Y'){
158         newState = q3;
159         toRight();
160     } else if(symbol == ' '){
161         newState = q4;
162         band[position] = 'B';
163         toRight();
164     } else {
165         System.err.println("q3 Cadena áinvlida");
166         file.closeFile();
167         System.exit(0);
168     }
169     file.writeSth(actualState + " - " + symbol + " ->" + newState);
170     return newState;
171 }
172
173 private void toLeft(){
174     position -= 1;
175 }
176
177 private void toRight(){
178     position++;
179 }
180
181 private void bandToFile(){
182     String str = "|";
183     for(int i = 0; i < band.length-1; i++)
184         str = str + band[i] + "|";
185     file.writeSth(str);
186 }
187 }

```