UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Lenguajes Formales y Autómatas

Grupo: 02 - Semestre: 2021-1

Tarea 4:
Autómatas Finitos

FECHA DE ENTREGA: 04/11/2020

Alumno:

Téllez González Jorge Luis

Álvarez Sánchez Miranda

1. Ejercicio 1

Considere el alfabeto {1,2,x,=}. Diseñe en JFLAP un autómata finito determinista que reconozca expresiones de multiplicación bien formadas como las siguientes:

1x2=1

21x1=22

11x2x1x2112=212

Figura 1: *Cadenas de prueba aceptadas.*

Y rechaze expresiones las mal formadas, como por ejemplo:

1x

1x2=

2x1x211

X2=3

Figura 2: *Cadenas de prueba rechazadas.*

Pruebe su autómata con el archivo CadenasEj1.txt y adjunte las capturas de pantalla correspondientes.

1.1. Resolución

El autómata diseñado en JFlap para la resolución de este ejercicio es el que puede observarse en la siguiente página. Para su realización se consideraron los siguientes puntos:

- Se mantuvo el orden estricto entre números y símbolos, pero abriendo la posibilidad a que cada número pudiera ser una secuencia cualquiera de 1's y 2's sin restricción alguna.
- Cada estado contiene bucles propios y transiciones múltiples para permitir todos los posibles casos de multiplicación entre 2 o más números posibles sin romper el orden de escritura.
- Para llegar al estado final, se debe, estrictamente pasar por q3 a q4 y de q4 a q5; q5 es el estado final, en este punto se tiene por lo menos un digito (1 o 2) y además planteamos la posibilidad de más dígitos.

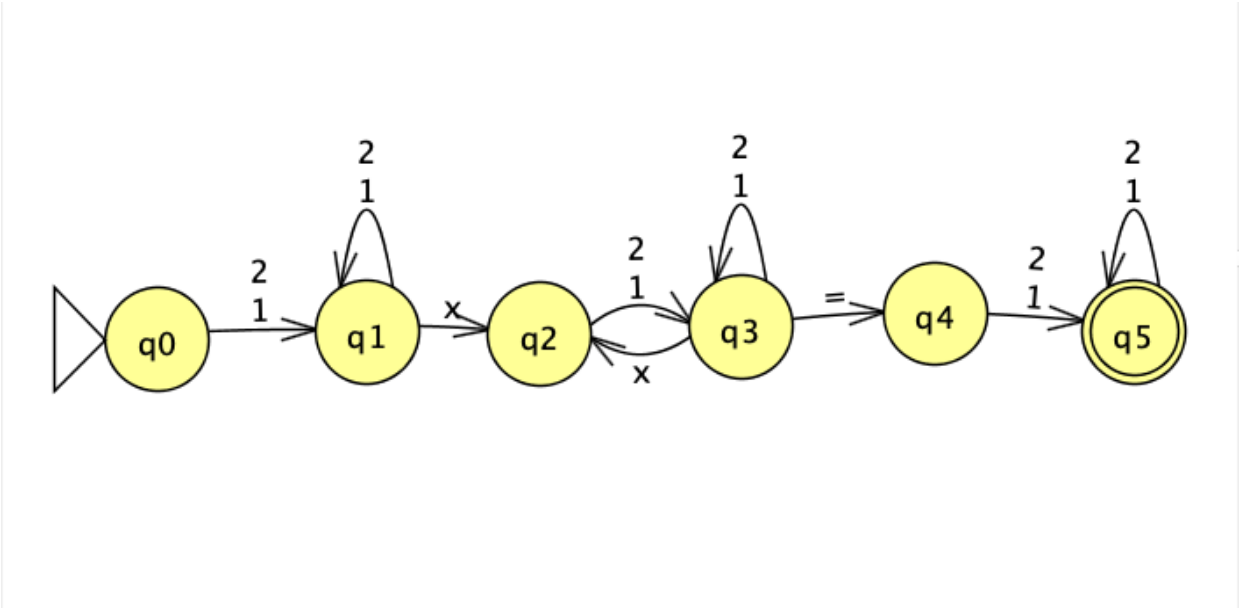

Figura 3: *Autómata del Ejercicio 1.*

Este autómata se puso a prueba con el archivo proporcionado previamente, arrojando los siguientes resultados que pueden verse en la siguiente página del documento:

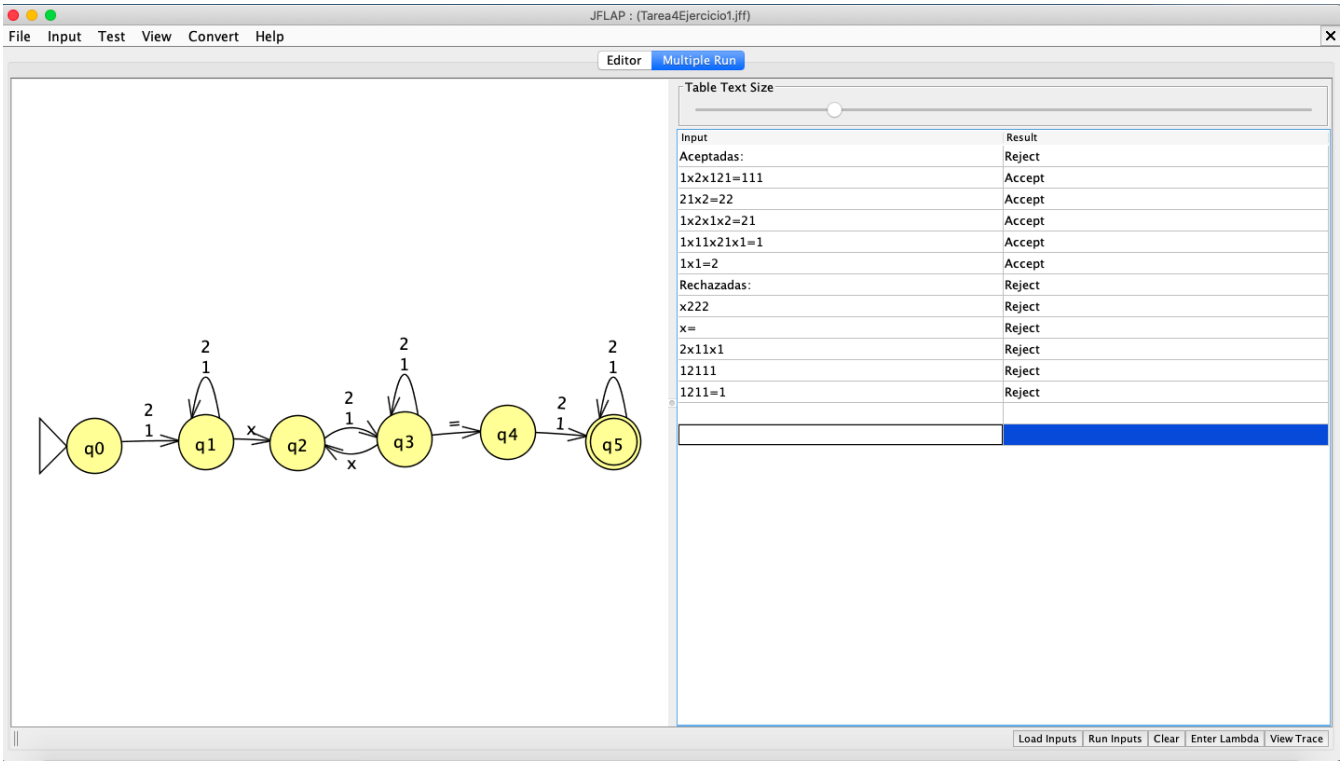

Figura 4: *Autómata del Ejercicio 1 puesto a prueba.*

Con lo anterior, el autómata generado cumple con las especificaciones solicitadas.

2. Ejercicio 2

En java los comentarios de parrafo se delimitan entre las secuencias / y */. El comentario puede tener cualquier secuencia de símbolos, excepto los delimitadores de comentario /* y */. Por simplicidad vamos a trabajar con el alfabeto {a, b, *, /}, de esta forma, algunos ejemplos de comentarios bien formados serían:*

```
/**/
/*****/
/*abbabb*/
/*ababa**/
/***aa**ab///b*/
```

Figura 5: *Cadenas de prueba aceptadas.*

y ejemplos de comentarios mal formados:

aaba

ab*a/a

*/**/**

/*abaa**

a***/a**/**

//abb**/**

/aa**//a*a****/**

Figura 6: *Cadenas de prueba rechazadas.*

Diseñe en JFLAP un AFD que acepte secuencias de uno o más comentarios bien formados, es decir, que acepte secuencias como:

Figura 7: *Cadenas de prueba aceptadas.*

y rechaze secuencias como:

Figura 8: *Cadenas de prueba rechazadas.*

Pruebe su autómata con el archivo CadenasEj2.txt y adjunte las capturas de pantalla correspondientes.

2.1. Resolución

El autómata diseñado en JFlap para la resolución de este ejercicio es el que puede observarse a continuación:

Para la creación de este autómatas se consideraron los siguientes aspectos:

- Cada comentario inicia con un /* estrictamente, pero no puede ocurrir que aparezca un * y un / a mitad de una cadena, así que si se introduce en una cadena un * necesariamente habrá dos / para que la estructura del comentario no sea incorrecta.

- Se consideran múltiples transiciones considerando el alfabeto de 4 caracteres, siempre considerando la regla principal que valida a los comentarios. Con esto, es posible generar comentarios con los 4 caracteres a la mitad de cada cadena válida.
- Únicamente los estados q2 y q3 pueden avanzar hacia q5 y, posteriormente, dirigir al estado final q6 de tal forma que se fuerza que el final de la cadena contenga estrictamente letras y, finalmente, /*.

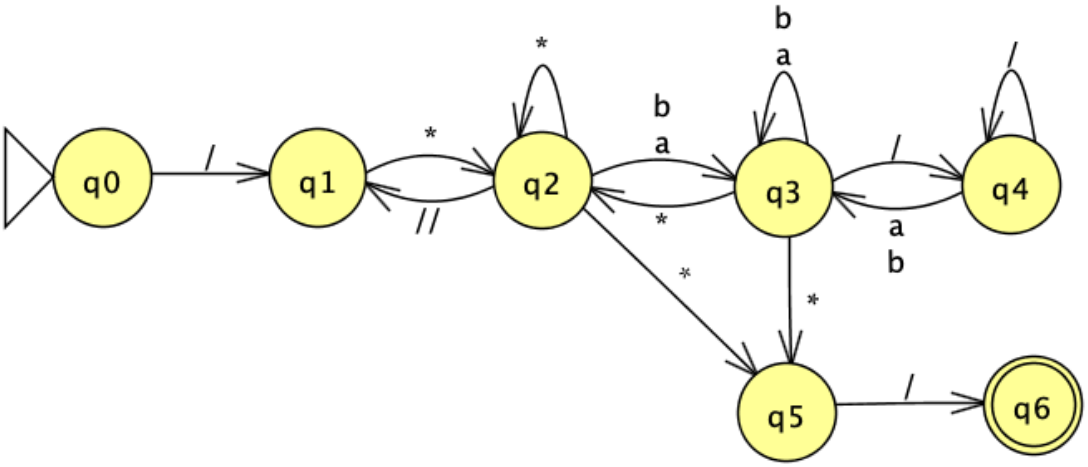

Figura 9: *Autómata del Ejercicio 2.*

En la siguiente página se muestra la prueba realizada al autómata generado con el archivo de texto plano proporcionado:

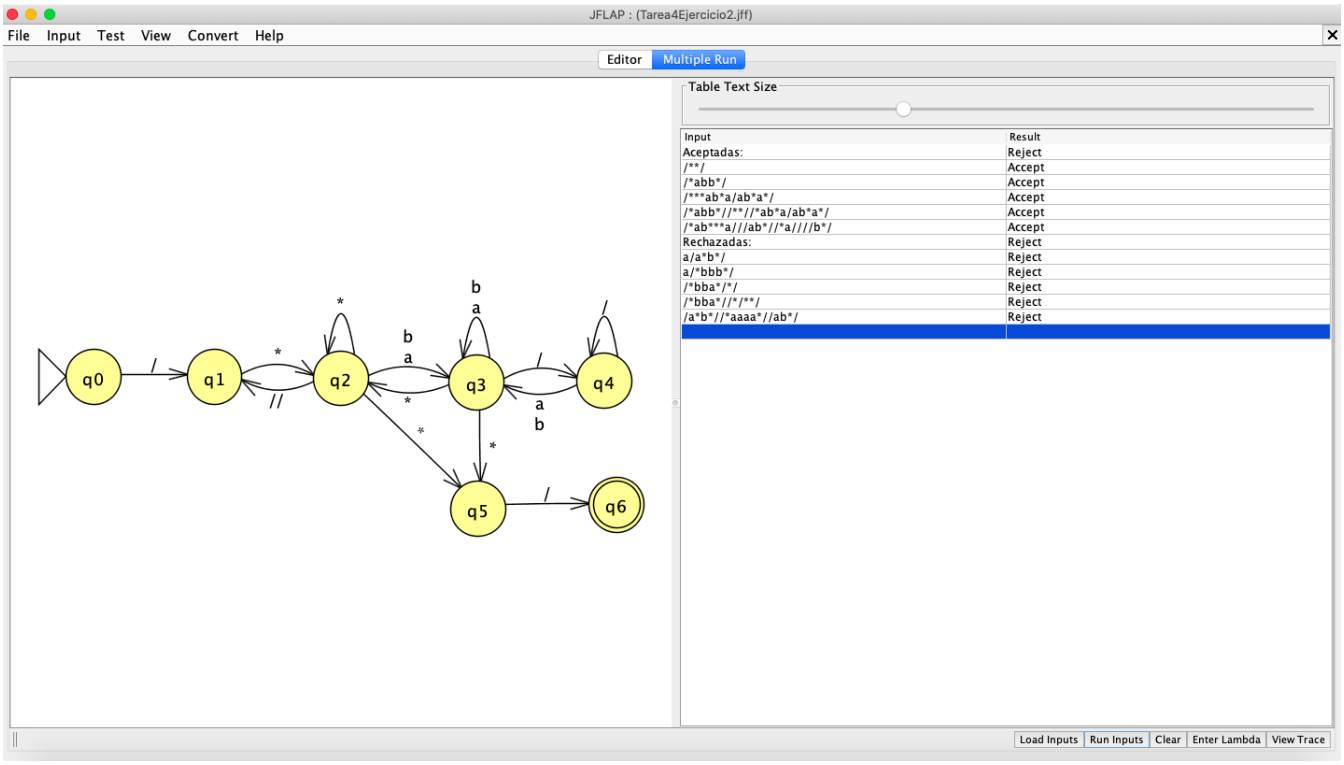

Figura 10: *Autómata del Ejercicio 2 puesto a prueba.*

3. Ejercicio 3

Utilice la plantilla Automata.java para programar el autómata diseñado en el ejercicio 1. Ponga a prueba su programación con las siguiente entradas y adjunte las capturas de pantalla.

```
1x21=2

1x1x2=11

1x2x2x11=21

1x

1x2=

=x==

2x1x2

x2x1=1
```

Figura 11: *Cadenas de prueba.*

3.1. Resolución

El código fue trabajado por medio de la plataforma **repl.it** y se hicieron modificaciones menores para que en el método principal aparecieran las cadenas y su mensaje de validación o rechazo, acorde a los métodos ya programados previamente. En las siguientes imágenes se muestran las salidas generadas por la plantilla modificada con la función delta de transición de estados programada con base en el autómata equivalente:

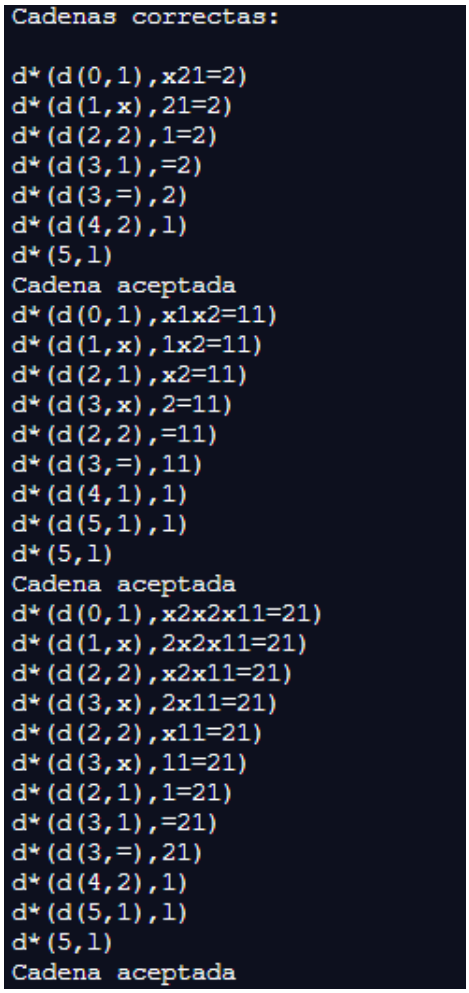

Figura 12: *Cadenas de prueba aceptadas por el autómata programado.*

```
Cadenas incorrectas:

d*(d(0,1),x)
d*(d(1,x),1)
d*(2,1)
Cadena rechazada
d*(d(0,1),x2=)
d*(d(1,x),2=)
d*(d(2,2),=)
d*(d(3,=),1)
d*(4,1)
Cadena rechazada
d*(d(0,=),x==)
d*(d(-1,x),==)
d*(d(0,=),=)
d*(d(-1,=),1)
d*(0,1)
Cadena rechazada
d*(d(0,2),x1x2)
d*(d(1,x),1x2)
d*(d(2,1),x2)
d*(d(3,x),2)
d*(d(2,2),1)
d*(3,1)
Cadena rechazada
d*(d(0,x),2x1=1)
d*(d(-1,2),x1=1)
d*(d(0,x),1=1)
d*(d(-1,1),=1)
d*(d(0,=),1)
d*(d(-1,1),1)
d*(0,1)
Cadena rechazada
Uso: java Automata <cadena de entrada>
```

Figura 13: *Cadenas de prueba rechazadas por el autómatas programado.*

Considerando que este autómata termina en q5, en el método **procesaCadena** se determinó que, cuando el entero *q* arrojado por **delta_extendida** sea q=5, la cadena sea aceptada (Es decir, logró llegar hasta el estado final). Caso contrario, será rechazada.

```
static void procesaCadena(String cadena){
  // q = d*(q0,w)
  int q = delta_extendida(0, cadena);
  if(q==5){
    System.out.println("Cadena aceptada");

  }else{
    System.out.println("Cadena rechazada");
  }

}
```

Figura 14: *Método procesaCadena.*