PRACTICA 1: EXPRESIONES REGULARES

CURSO: 591000 - COMPILADORES
NOMBRE: JOSEPH PRINCIPE
APELLIDO: PAJUELO CONDORI

ÍNDICE:

1.EXPRESIÓN REGULAR:

Las expresiones regulares implementadas en la práctica son las siguientes:

1. **x*(z | y)*x+**
2. **z+y?(x | z)z***

Alfabeto implementado: {x,yz}
Explicación de las expresiones regulares:

1.- **x*(z | y)*x+**
x* -> La cadena de texto debe contener la letra “x” cero o más veces.
(z | y)* -> La cadena de texto debe contener la letra “z” o “y” cero o más veces.
x+ -> La cadena de texto debe contener la letra “x” una o más veces.

2.- **z+y?(x | z)z***
z+ -> La cadena de texto debe contener la letra “z” una o más veces.
y? -> La cadena de texto debe puede o no tener la letra “y”.
(x | z) ->La cadena de texto debe contener la letra “x” o “z”.
z* -> La cadena de texto debe contener la letra “z” cero o más veces.

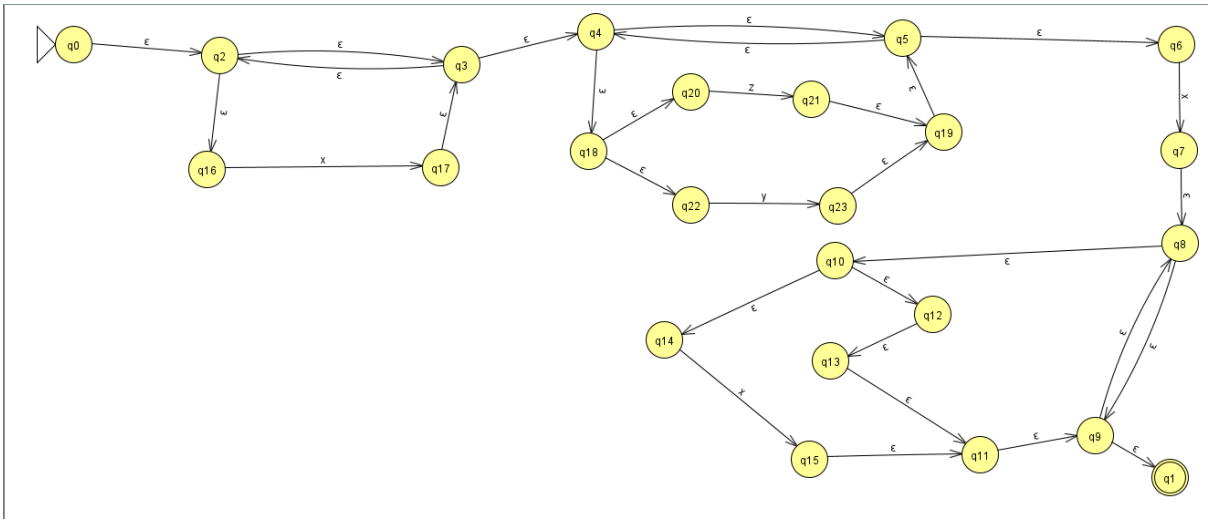
2.TRANSFORMACIÓN DE ER A JFLAP:

Para poder introducir a JFLAP expresiones regulares primero debemos realizar ciertas transformaciones debido a que JFLAP no acepta y mal interpreta ciertos caracteres.
En JFLAP el carácter de “|” se implementa como “+”, en caso de implementar un “?” debemos hacer la siguiente transformación (! + ‘’), se asocia un or vacío con el carácter en cuestión que esté implementando.
En caso de utilizar “+” la transformación es la misma que con el “?”, simplemente debemos asegurarnos que se implemente como mínimo una vez el carácter en cuestión ej.
ER: a+ -> JFLAP: a(! + a)

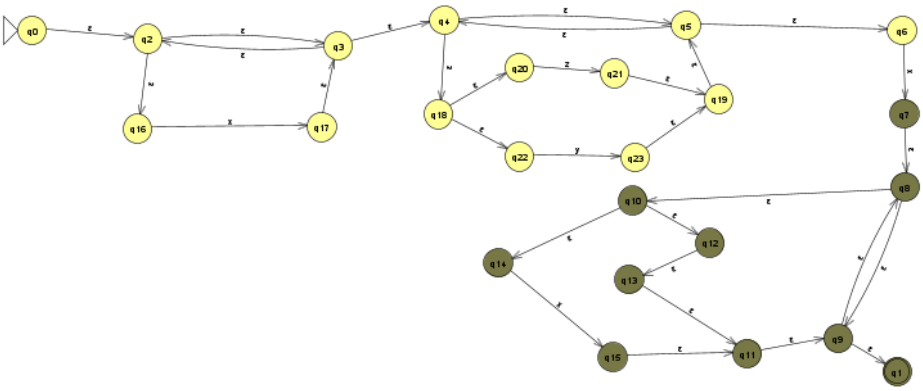
1. x*(z + y)*x(! + x)*
2. z(! + z)*(! + y)(x + z)z*

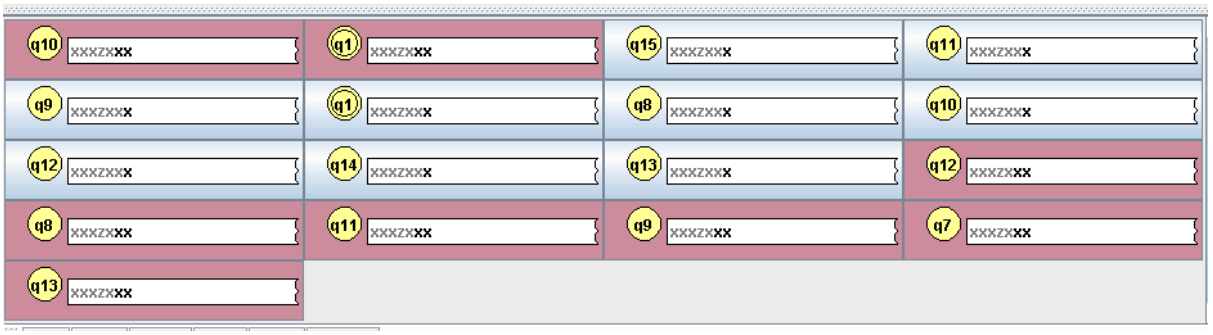
3.COMPROBACIÓN AUTÓMATA FINITO NO DETERMINISTA:

Introducimos en JFLAP nuestra expresión regular x*(z + y)*x(! + x)* y verificamos los estados creados.

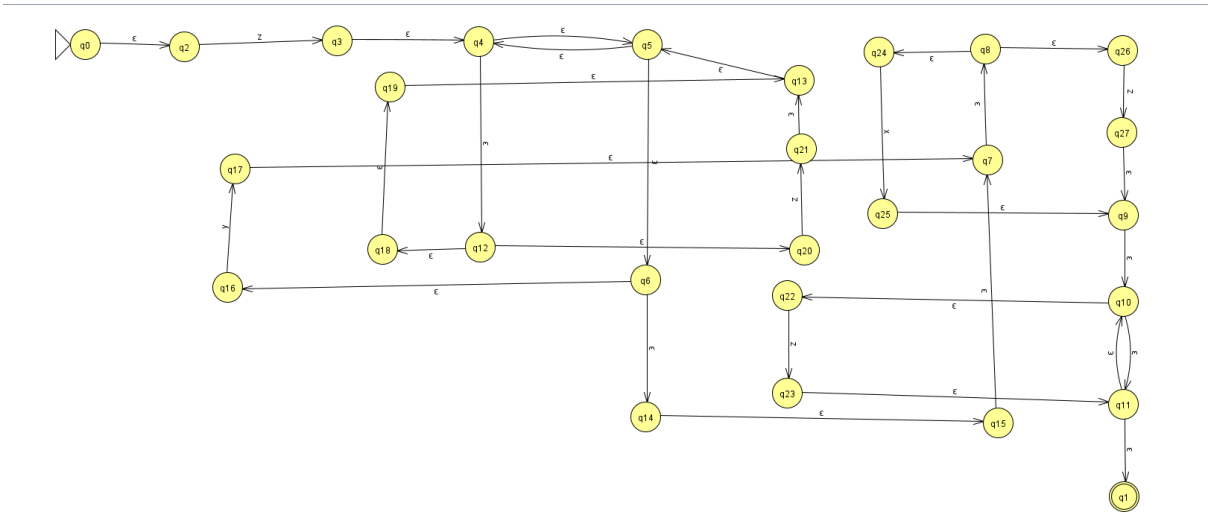


Realizamos un input de “xxxzxxx” para comprobar al automata.

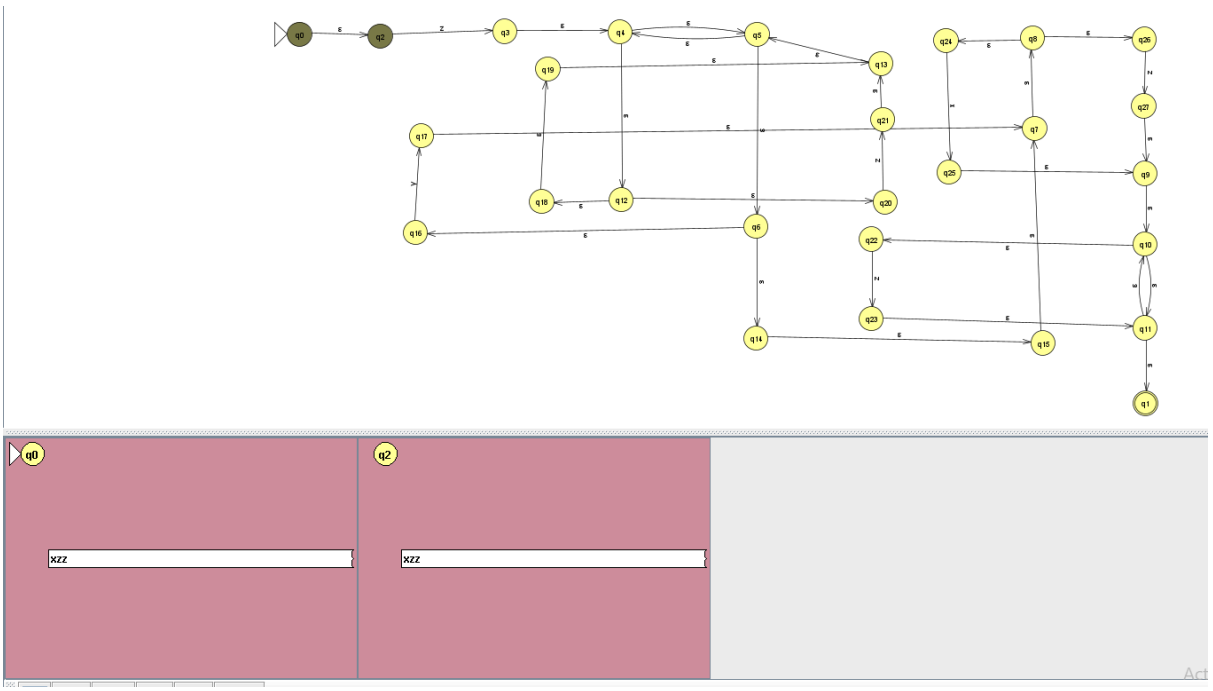




Introducimos en JFLAP nuestra expresión regular z(! + z)*(! + y)(x + z)z* y verificamos los estados creados.

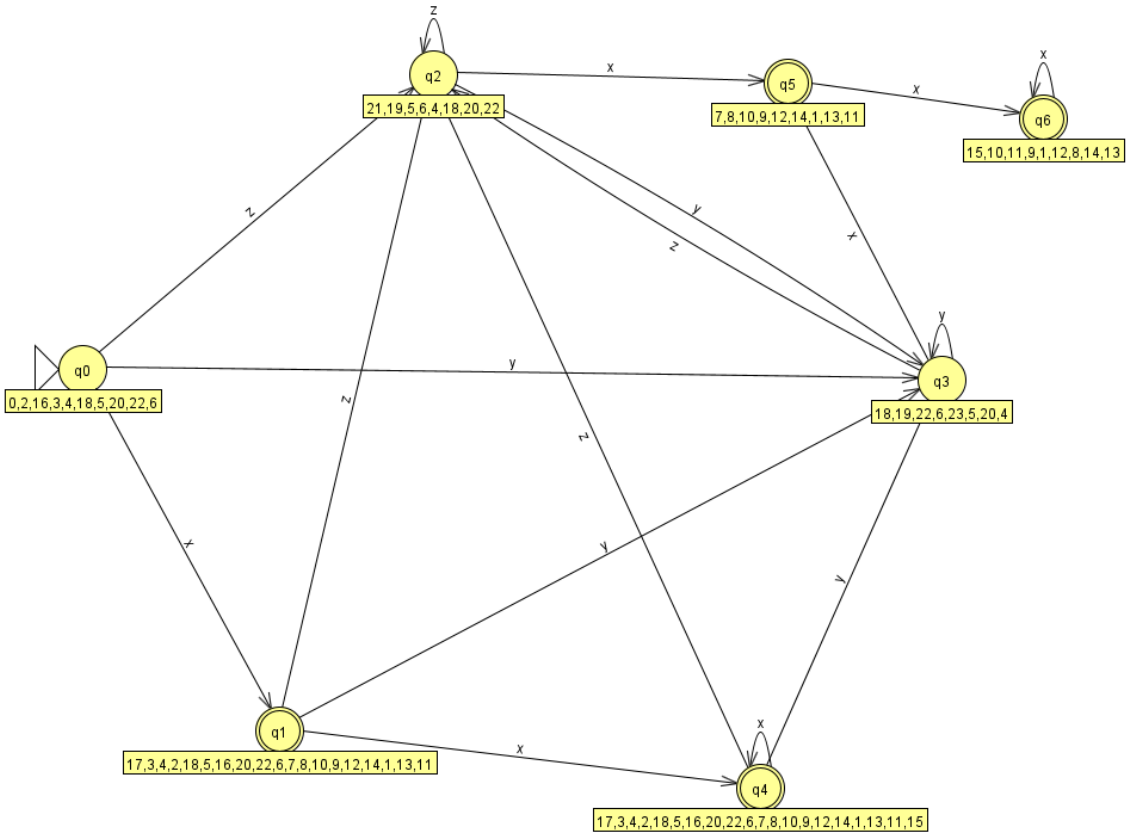


Comprobamos el autómata finito no determinista.

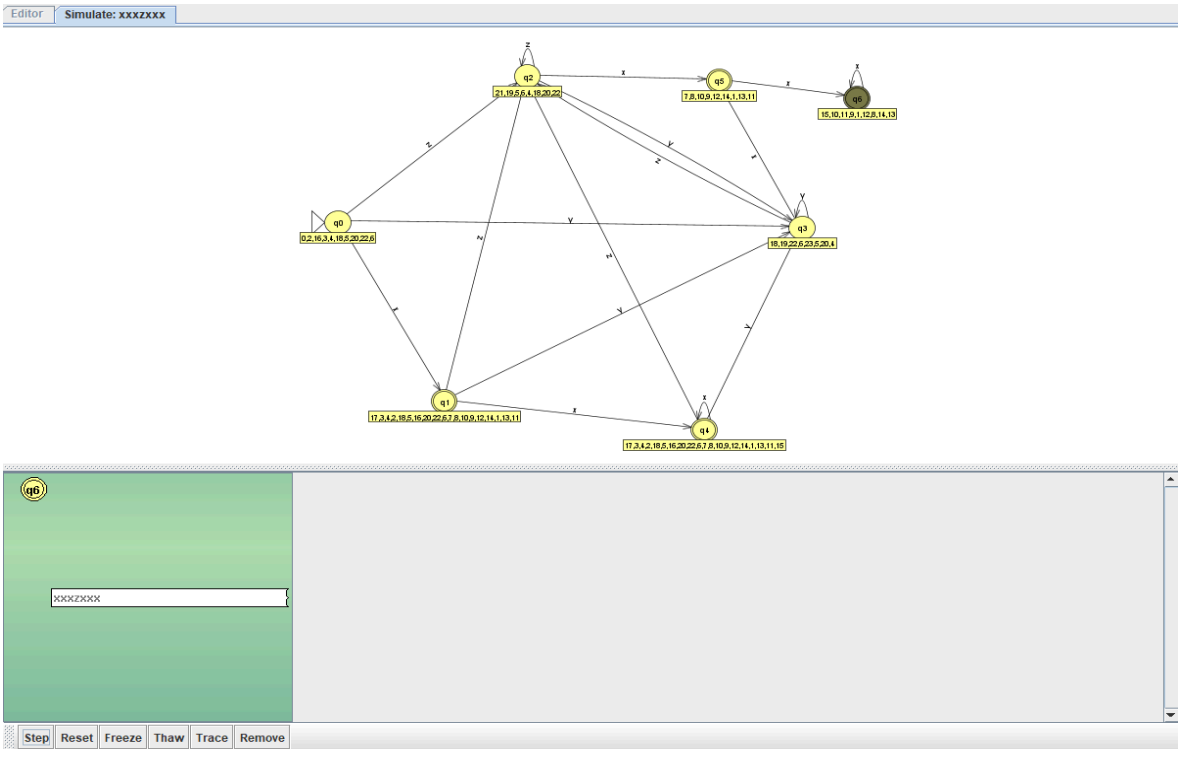


4.TRANSFORMACIÓN AUTÓMATA FINITO DETERMINISTA

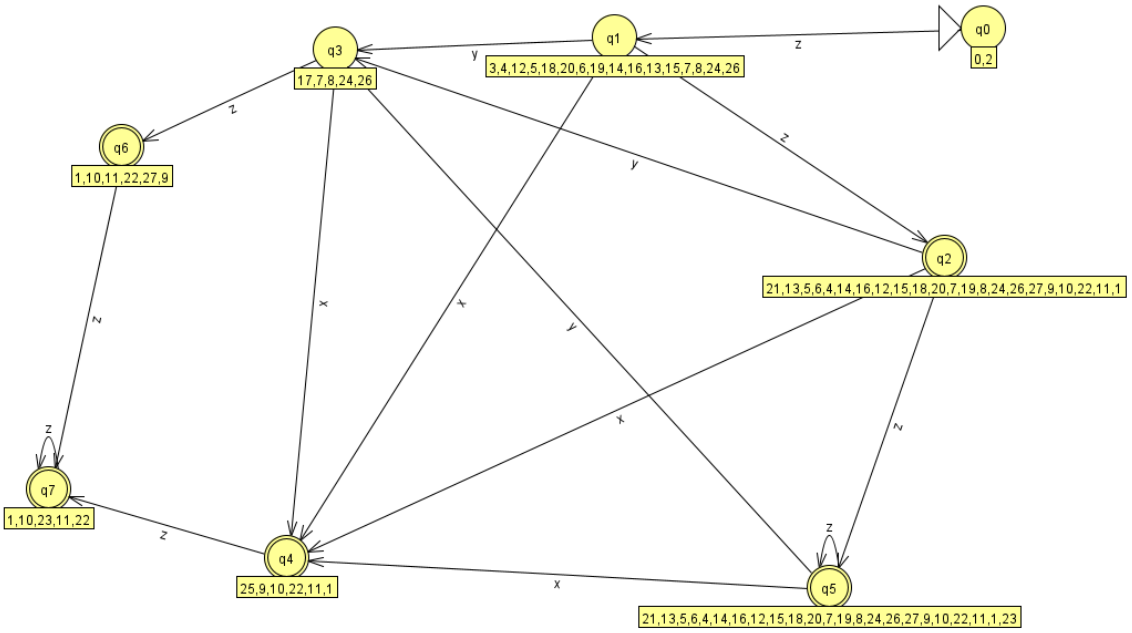
Mediante el autómata finito no determinista, creamos el autómata finito determinista.

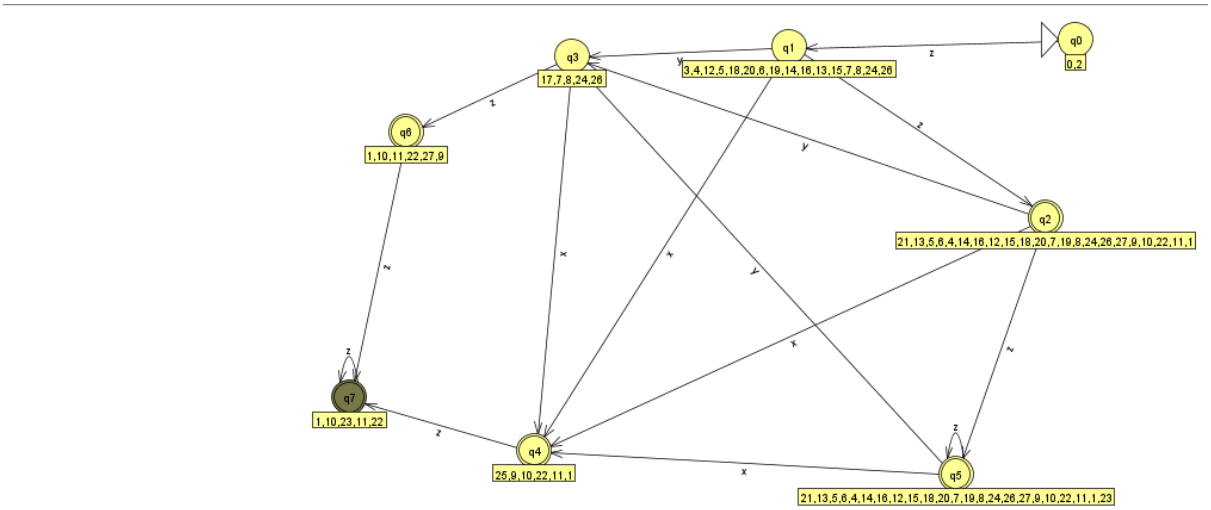


Realizamos un input de la cadena “xxxyxxx” para comprobar que el autómata detecta, si la cadena es válida con respecto a nuestra expresión regular x*(z + y)*x(! + x)*.



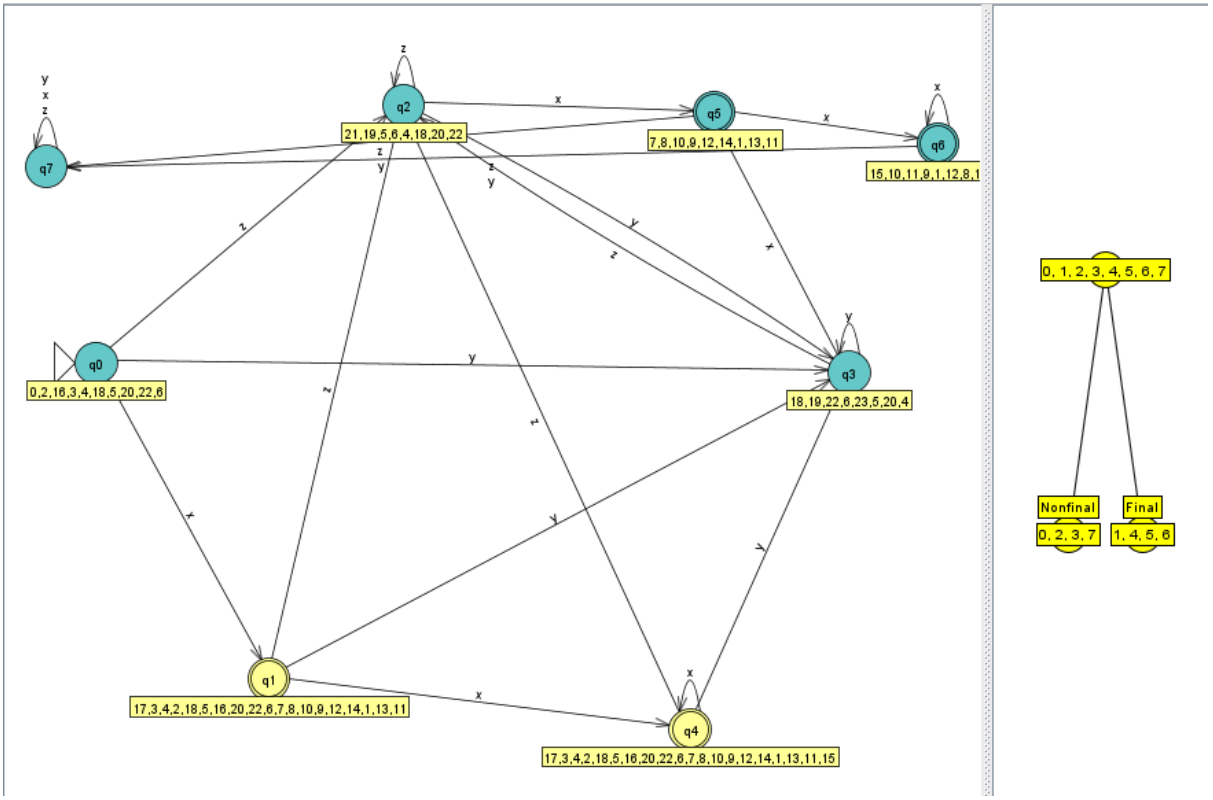
Pasamos al autómata finito determinista de z(! + z)*(! + y)(x + z)z*



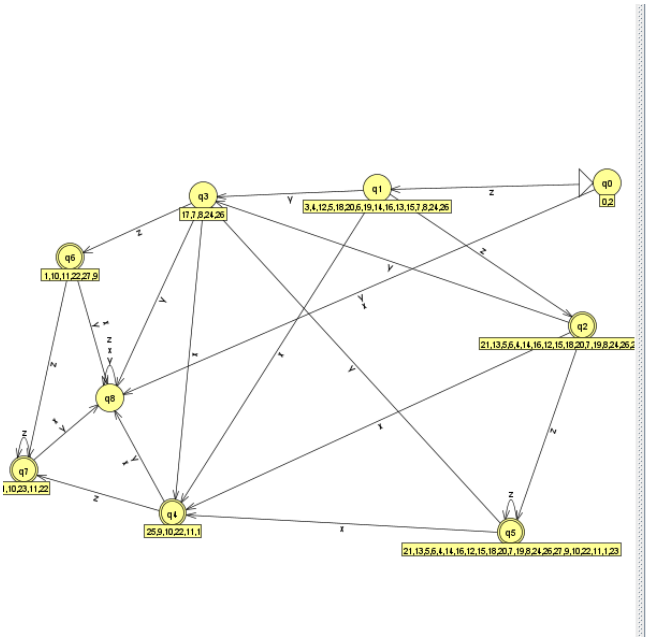


5.MINIMIZACIÓN DEL AUTÓMATA FINITO DETERMINISTA

x*(z + y)*x(! + x)*

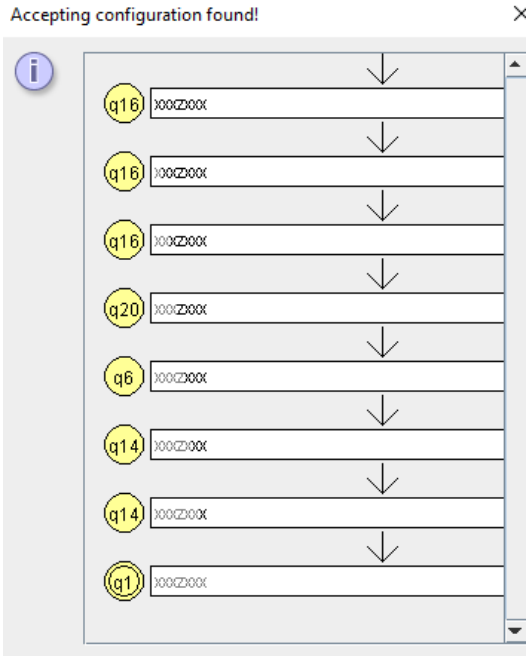


z(! + z)*(! + y)(x + z)z*

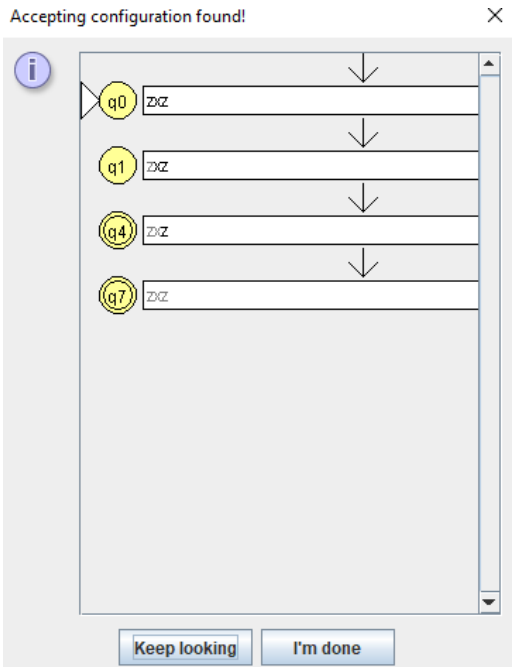


6.COMPROBACIÓN DEL AUTÓMATA FINITO DETERMINISTA MINIMIZADO

x*(z + y)*x(! + x)*



z(! + z)*(! + y)(x + z)z*



7.MATRIZ DE ESTADOS

1.x*(z + y)*x(! + x)*

	x	y	z
q0	1	3	2
q1	4	3	2
q2	5	3	2
q3	5	3	2
q4	4	3	2
q5	6	0	0
q6F	6	0	0

2.z(! + z)*(! + y)(x + z)z*

	x	y	z
q0I	0	0	1
q1	4	3	2
q2F	4	3	5
q3	4	5	6
q4F	0	0	7
q5F	4	3	5
q6F	0	0	7
q7F	0	0	7

8.CÓDIGO MATRIZ DE ESTADOS:

Propósito: El programa implementa una matriz de estados para la comprobación de cadenas de caracteres, tanto para sí son válidas o no.
Tecnologías implementadas: Java, IntelliJ IDEA, librería java.util.*.
Estructura: El codigo esta compuesto por dos clases principales:
La clase Automata es la que carga con todo la lógica e implementación de la matriz de estados.
La clase InputType se encarga de la creación de las 100 cadenas de caracteres e implementa la comprobación de la clase autómata para guardar las cadenas válidas en una lista.
La matriz de estados es creada a través de enlazar un hashmap principal que tendrá los estados con otros hashmap que contendrán las trazas y se irán uniendo de acuerdo al número de estados por orden.
Métodos usados para la implementación de la matriz de estados: crearMatriz, creadorTraza.
Funcionamiento del programa:
El programa inicia pidiendo los datos necesarios para la creación de la matriz de estados: Estado inicial, estados finales, los estados, el alfabeto implementado y las trazas de cada estado.Después de introducir estos datos, se le pedirá al usuario introducir la cadena de caracteres a comprobar. Por último el programa le dirá si es correcta o no la cadena en cuestión.
En caso de querer hacer una comprobación manual, se debe descomentar la última parte comentada además esto es necesario en caso de querer probar las 100 cadenas aleatorias.

```
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q2 separada por comas (ejemplo: 0,0,1): 4,3,5
Como ingresar la traza: qo: x y z -> 0,0,1
Las posiciones respetan el orden del alfabeto, en caso de que el estado no tenga una conexion con otro se colocara un 0
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q3 separada por comas (ejemplo: 0,0,1): 4,5,6
Como ingresar la traza: qo: x y z -> 0,0,1
Las posiciones respetan el orden del alfabeto, en caso de que el estado no tenga una conexion con otro se colocara un 0
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q4 separada por comas (ejemplo: 0,0,1): 0,0,7
Como ingresar la traza: qo: x y z -> 0,0,1
Las posiciones respetan el orden del alfabeto, en caso de que el estado no tenga una conexion con otro se colocara un 0
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q5 separada por comas (ejemplo: 0,0,1): 4,3,5
Como ingresar la traza: qo: x y z -> 0,0,1
Las posiciones respetan el orden del alfabeto, en caso de que el estado no tenga una conexion con otro se colocara un 0
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q6 separada por comas (ejemplo: 0,0,1): 0,0,7
Como ingresar la traza: qo: x y z -> 0,0,1
Las posiciones respetan el orden del alfabeto, en caso de que el estado no tenga una conexion con otro se colocara un 0
Si tiene un salto a otro estado se colocara el numero del estado q1 -> 1
ejemplo: q0 salta a q1 por z y no tiene saltos por x ni y -> 0,0,1
Ingrese la traza q7 separada por comas (ejemplo: 0,0,1): 0,0,7
Ingrese la cadena a evaluar (ejemplo: zyxzzzzzz): zyxzzzz
es correcto
```