Ejercicio 2. Encontrar un autómata que acepte el mismo lenguaje que el asociado a la expresión regular $(0+10)^{*}011$

PRIMERA PARTE Expresión Regular → Autómata

El autómata tiene que aceptar cadenas que contengan previamente las subcadenas 0 o 10 en cualquier número (* → >= 0) y orden (no tienen que venir primero todos los 0 y después todos los 10) y que la cadena termine en 011.

Ejemplo de cadena que se acepta: 0010010011

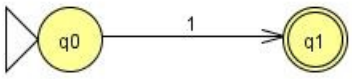
(***NOTA: no he seguido los pasos que seguiría jflap, me he ahorrado estados "inútiles para hacer esta demostración aunque sé que siguiendo el procedimiento del analizador léxico se generan esos estados (como q1) y que se eliminan minimizando.)

Cómo diría Jack el Destripador, vamos por partes:

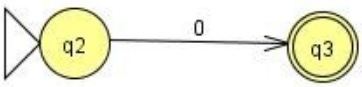
A) Construimos autómatas que acepten las subcadenas 0, 10 y 011

a.1) Autómata que acepta la cadena 10

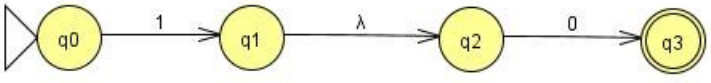
Autómata que acepta un 1



Autómata que acepta un 0

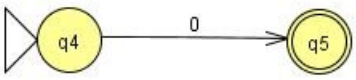


Para aceptar 10, uno ambos autómatas por transiciones nulas. Q1 pierde su carácter de estado final y Q2 pierde su carácter de estado inicial.

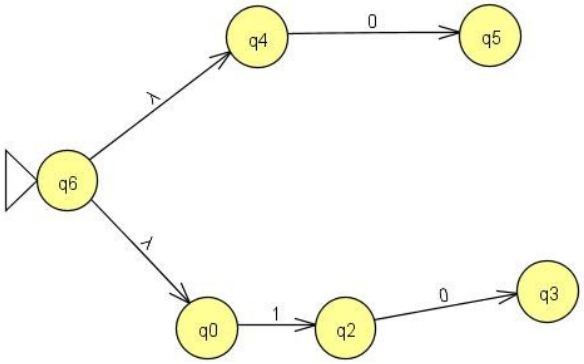


Q1 se puede suprimir y pasar directamente de q0 a q2 leyendo un 1.

a.2) Autómata que acepta la cadena 0

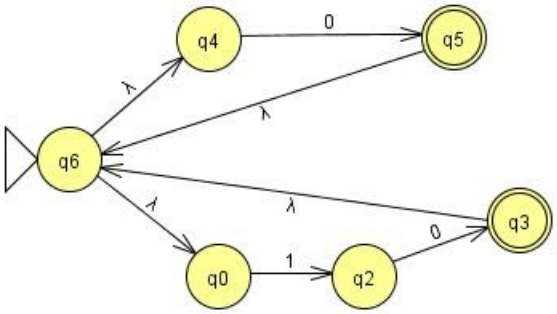


B) Ya tenemos los autómatas que aceptan las cadenas 10 y 0 por separado. Ahora crearemos un nuevo autómata que acepte 0 + 10 (ser capaz de leer 0 y aceptarlo o ser capaz de leer 10 y aceptarlo). Unimos el autómata final de a.1 y el autómata de a.2, para ello necesitaremos de un nuevo estado ‘Q6’ que servirá como estado inicial y que se conectará al autómata final de a.1 y al autómata final de a.2 por transiciones nulas.



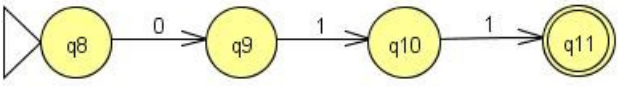
Q4 y Q0 pierden su carácter de estado final

b.1) Para conseguir que sea (0+10)* conectaremos q5 y q3 a q6 por transiciones nulas para que pueda leer tantas cadenas de 0 o 10 cómo sean necesarias (siguiendo los pasos del analizador léxico aparecería un nuevo estado inicial Q7 que se uniría a Q6 por transición nula y Q6 perdería su carácter inicial).

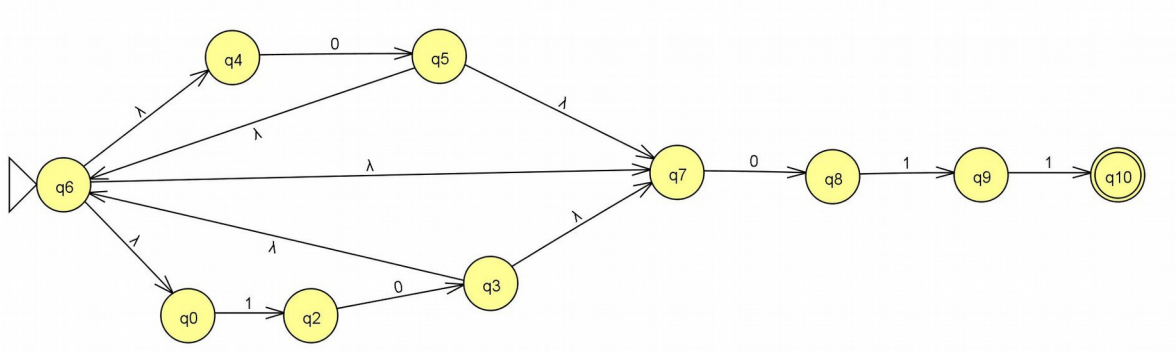


C) Ahora tenemos el autómata que acepta cadenas (0+10)*. Nos falta crear el autómata que acepta la cadena 011 y unirlo al autómata del b.1.

c.1) Autómata que acepta la cadena 011



c.2) Uno el autómata c.1 con el de b.1 por transiciones nulas



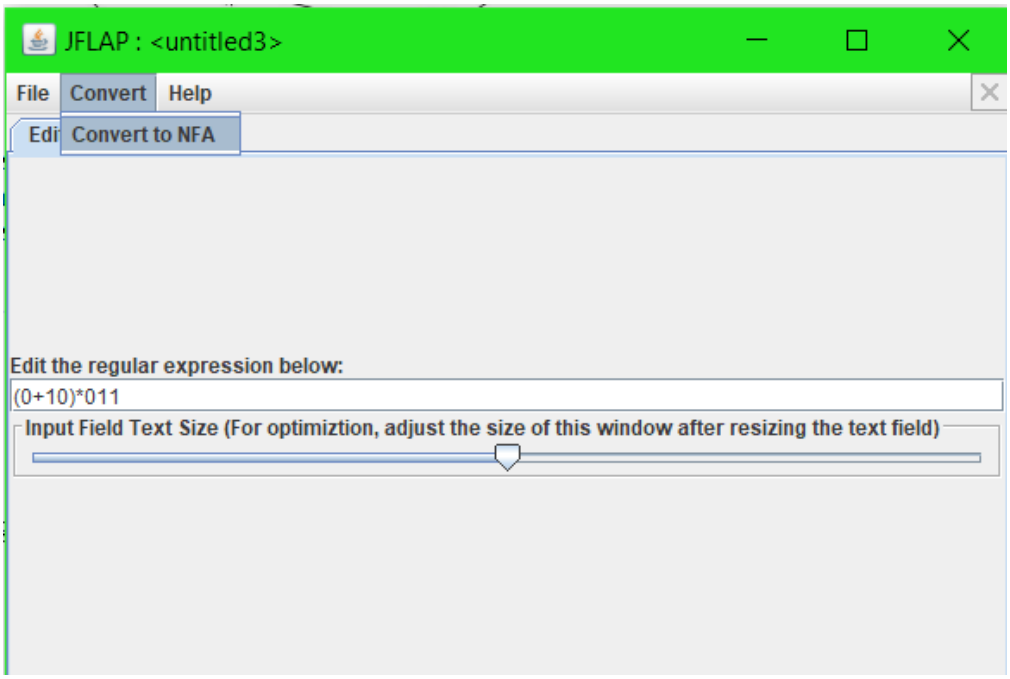
Conecto q6 (estado inicial) por transición nula a q7 dado que la cadena puede ser únicamente 011 (* → >= 0)
Conecto q5 y q3 a q7 dado que la cadena puede tener previamente cadenas 0 o 10.

Ya tengo el autómata que necesito para leer cadenas correspondientes a la expresión regular (0+10)*011.

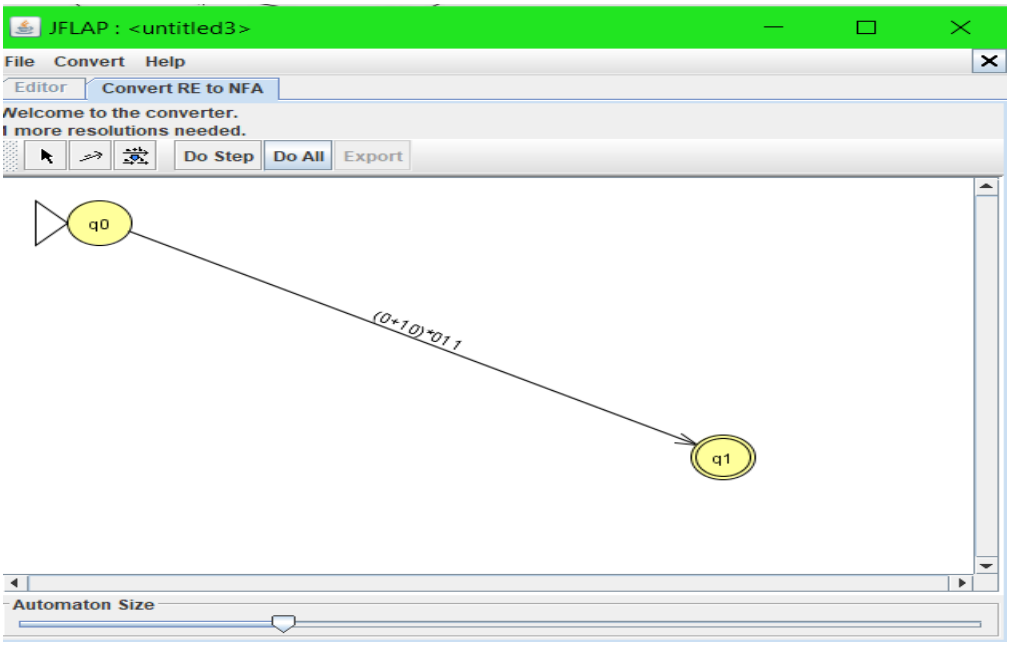
JFLAP

Me voy a regular expressions e introduzco (0+10)*011

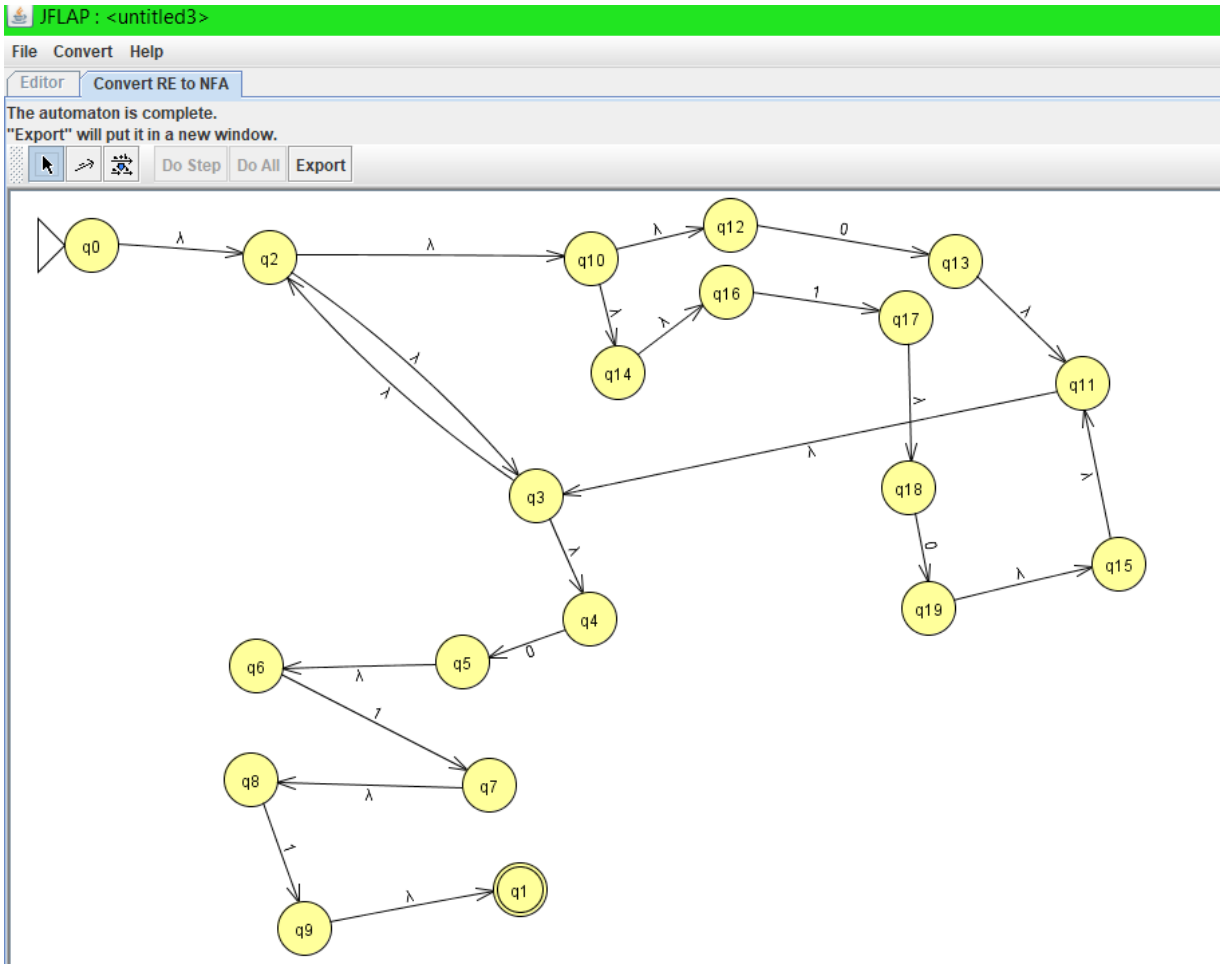
Lo convierto a autómata finito no determinista.



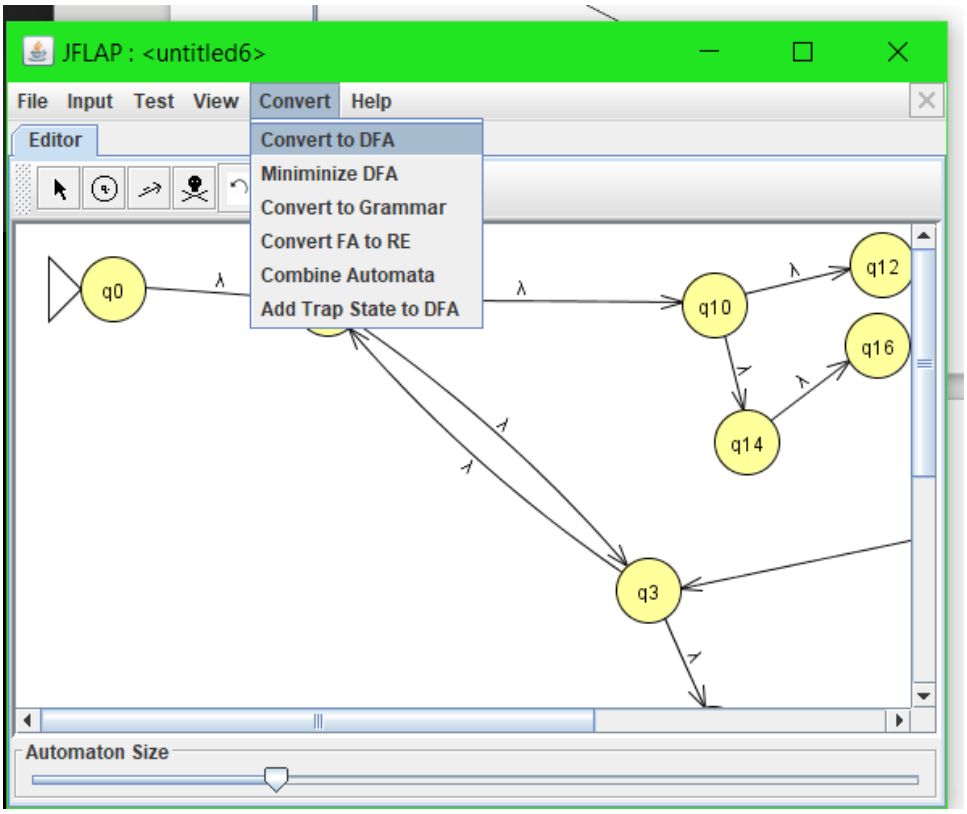
Me da como resultado el siguiente autómata:



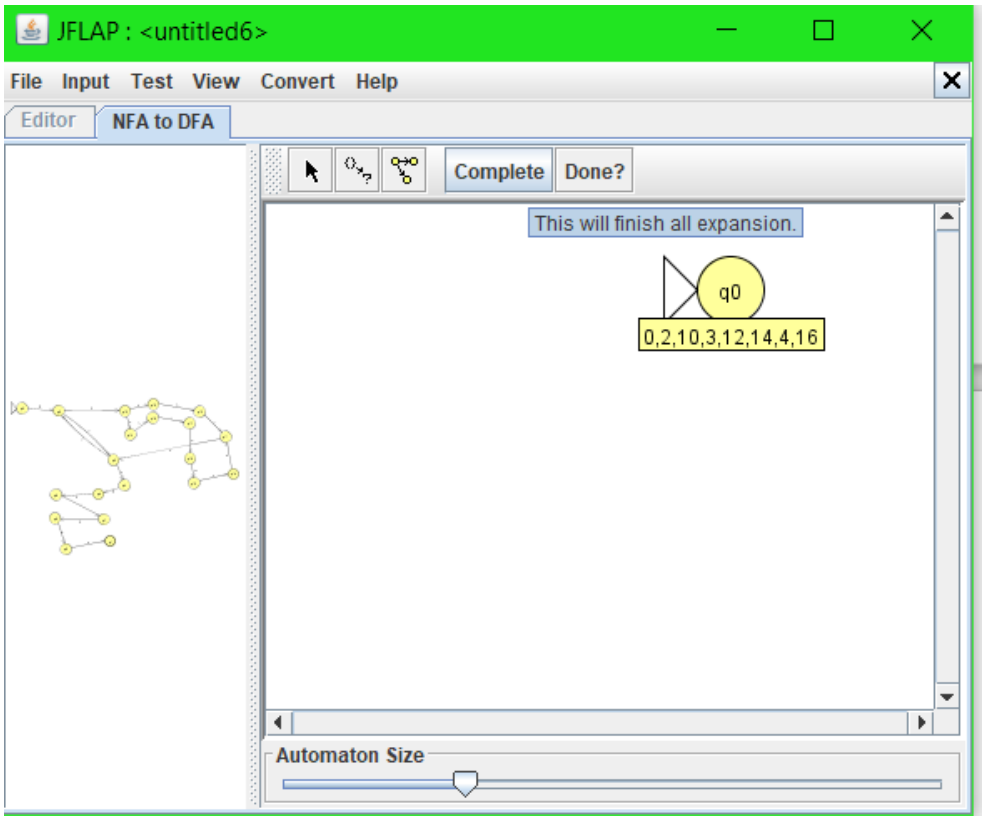
Do all → Me termina el autómata

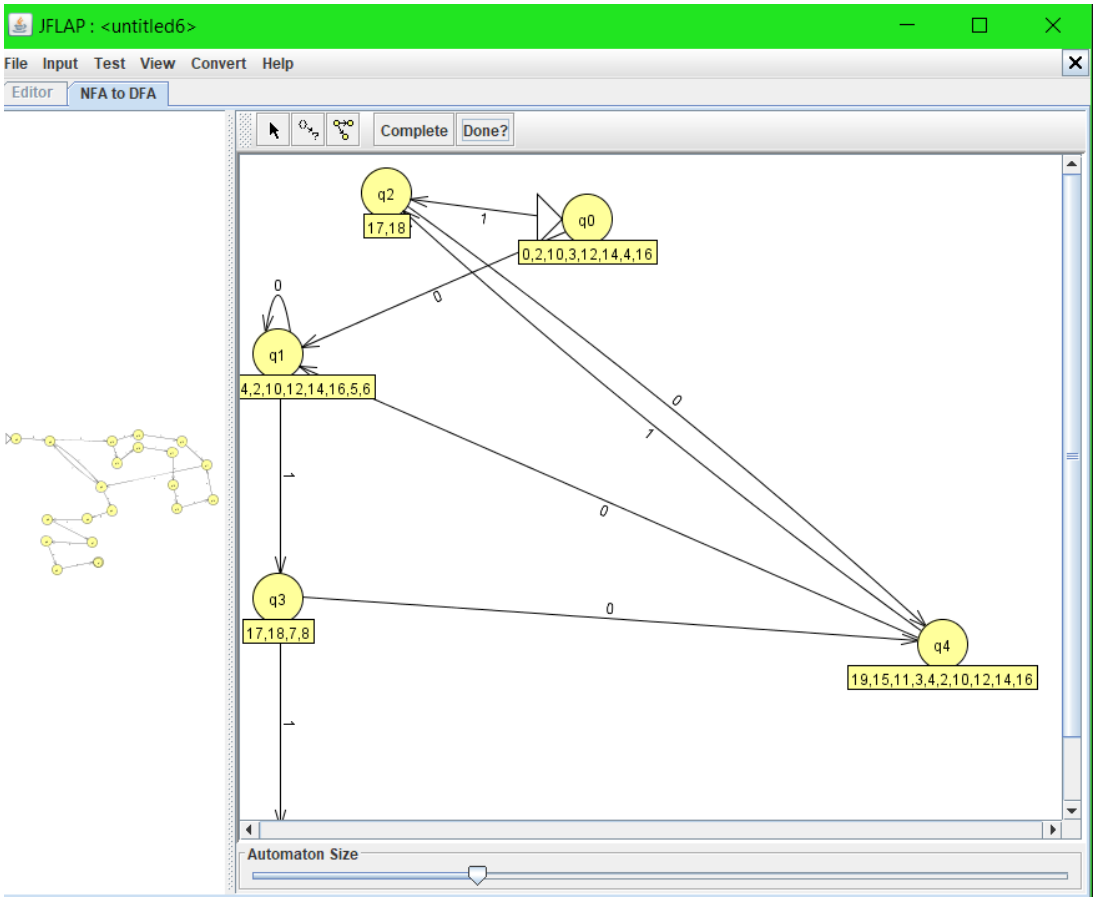


Lo exporto y paso a autómata determinista.

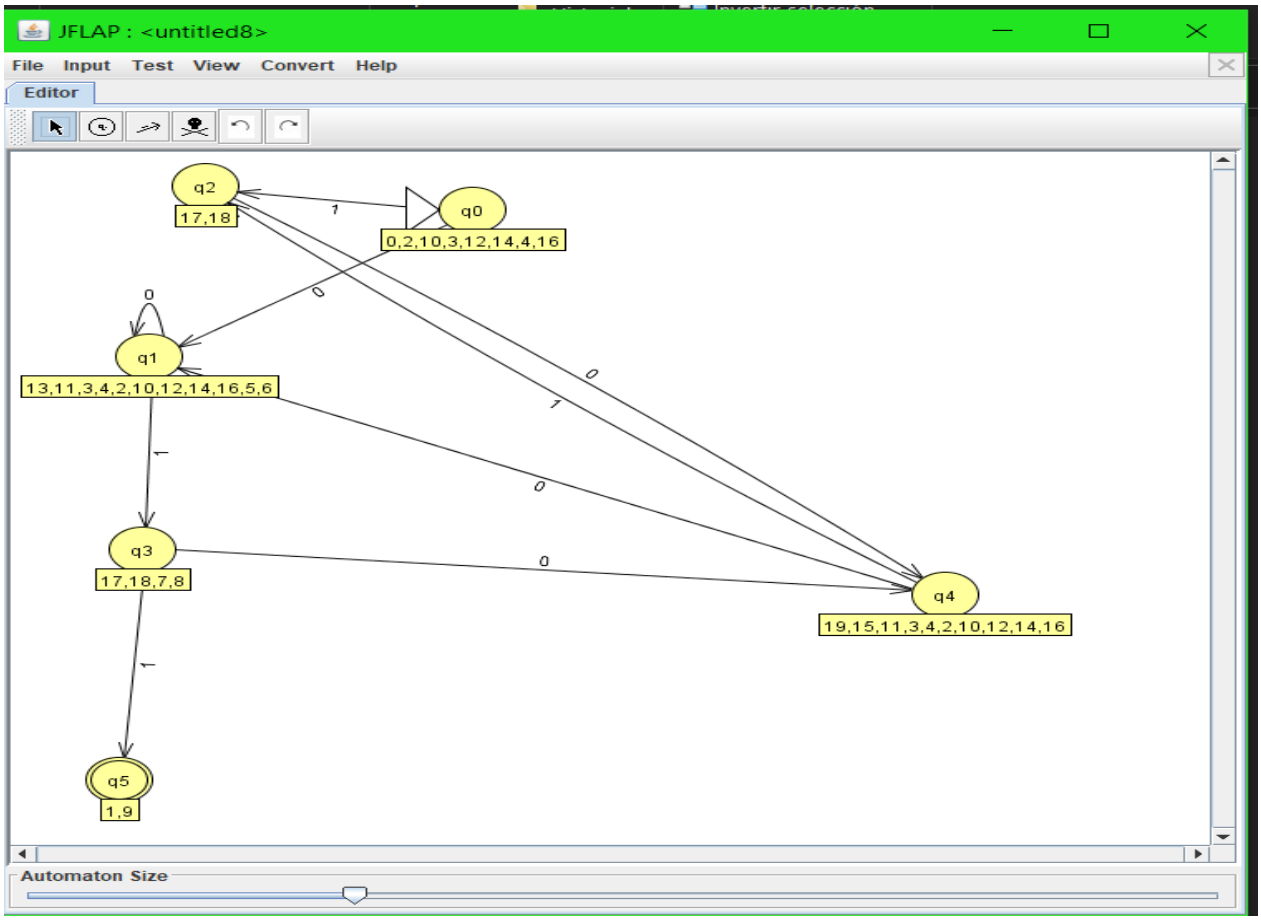


Le doy a complete

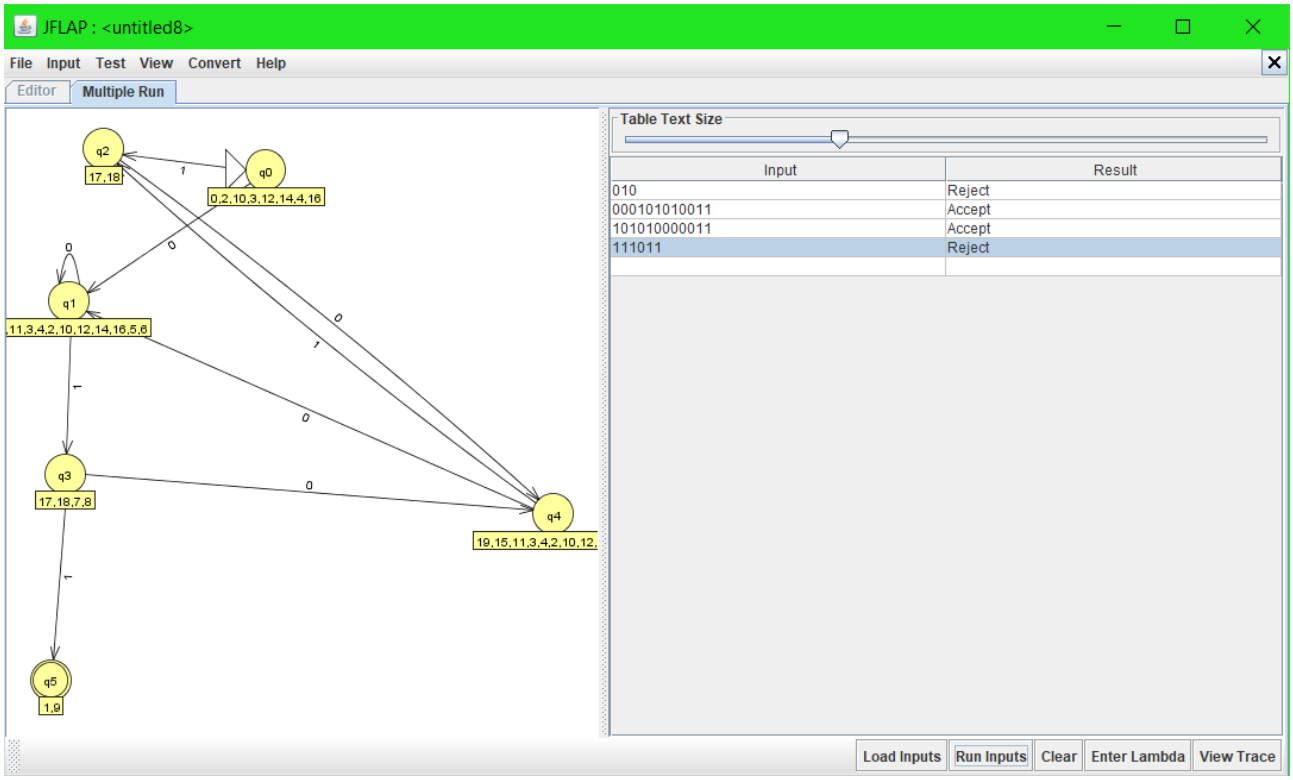




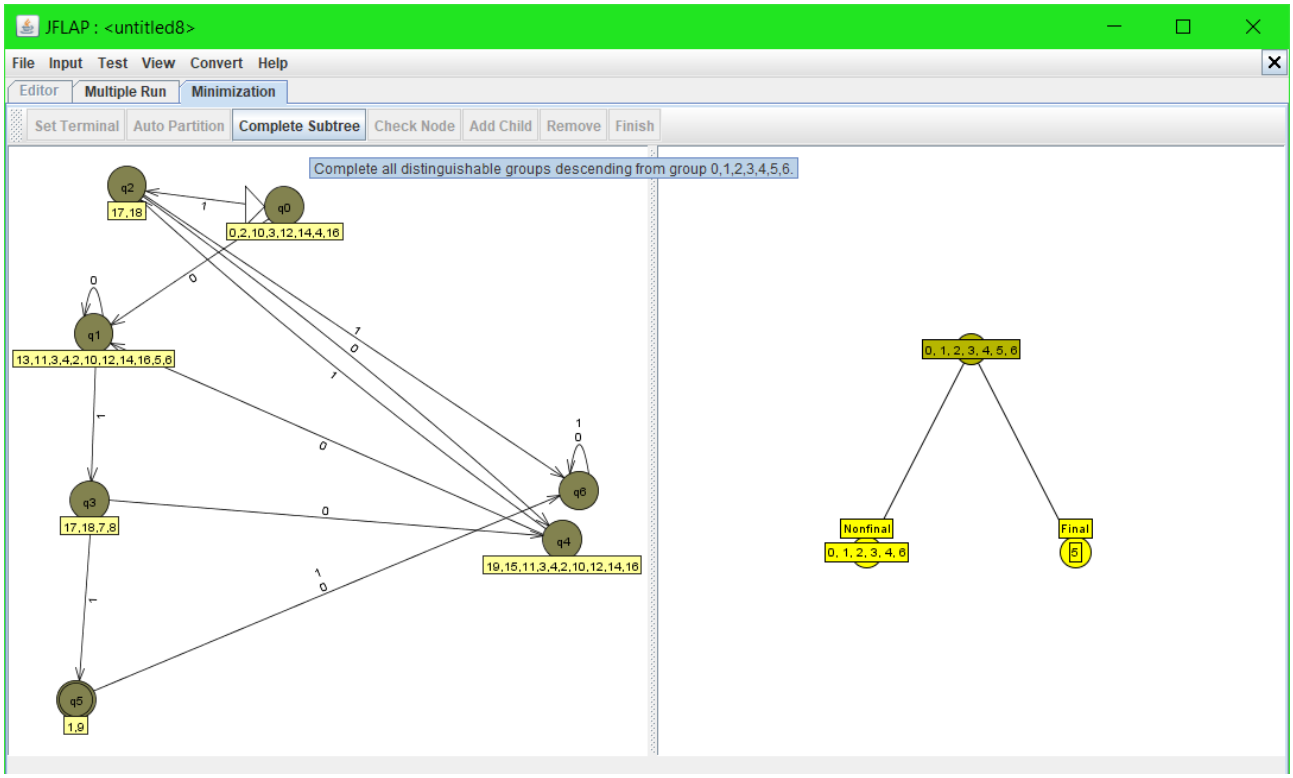
Le doy a done y se exporta. Ya tengo el autómata determinista.

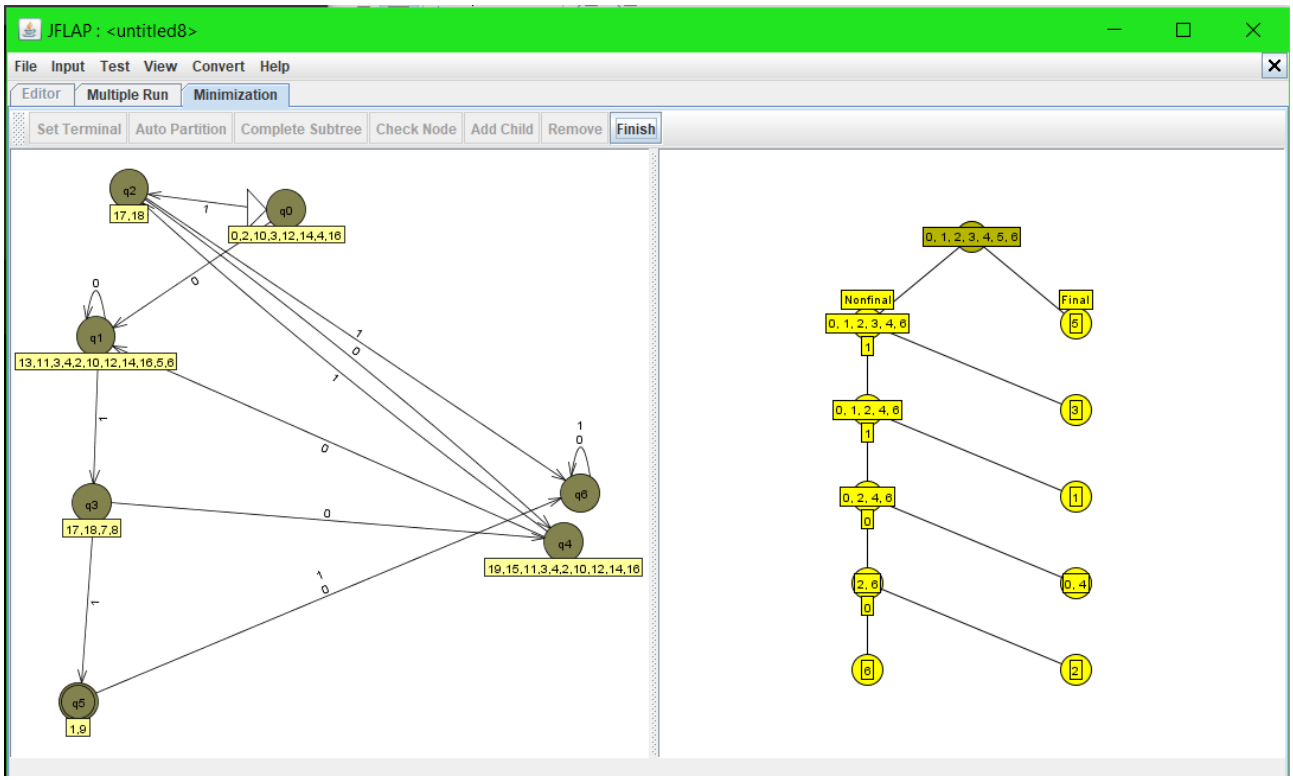


Ahora, voy a probar unas cuantas cadenas para ver si funciona correctamente.

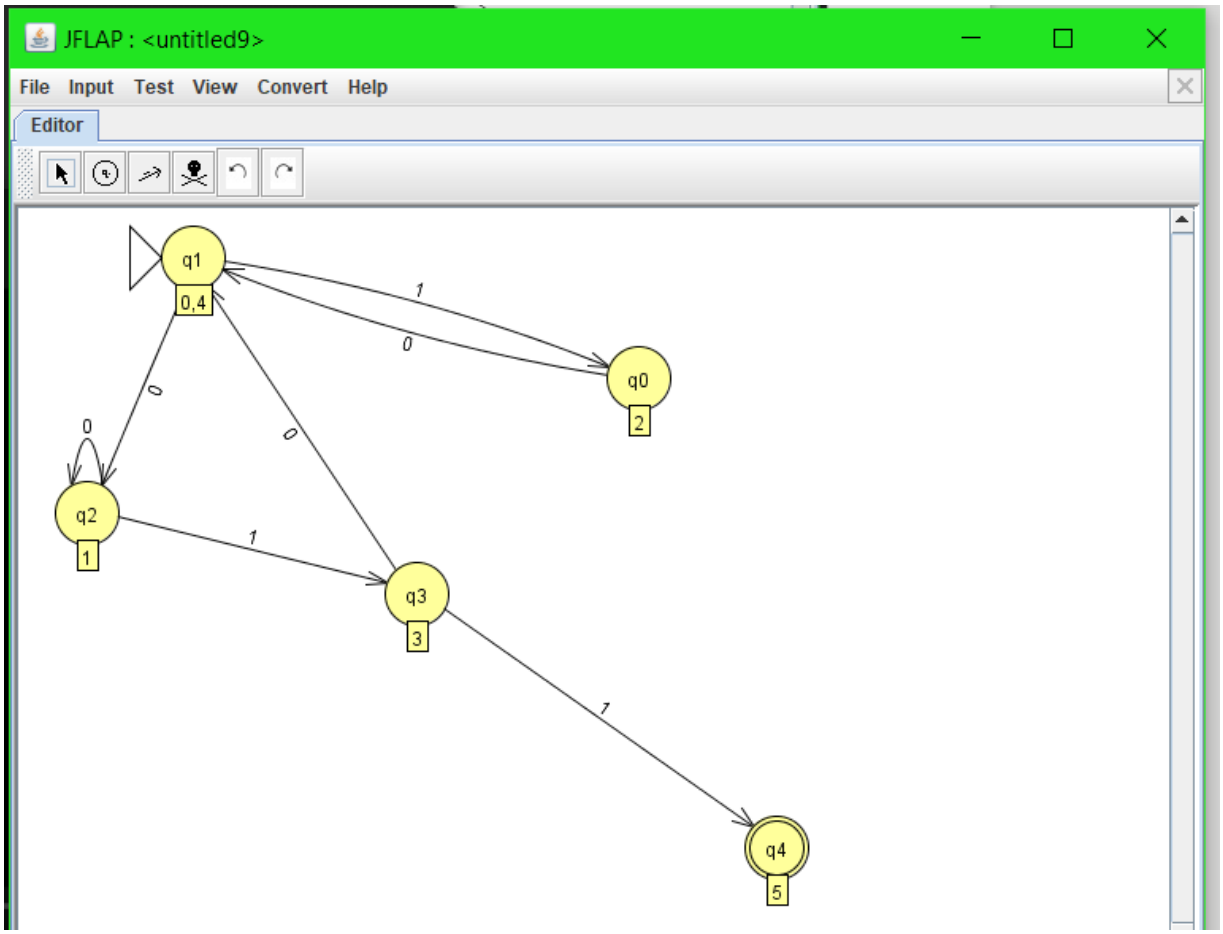


Parece que está todo en orden. Por último voy a mostrar como quedaría el autómata minimizado:



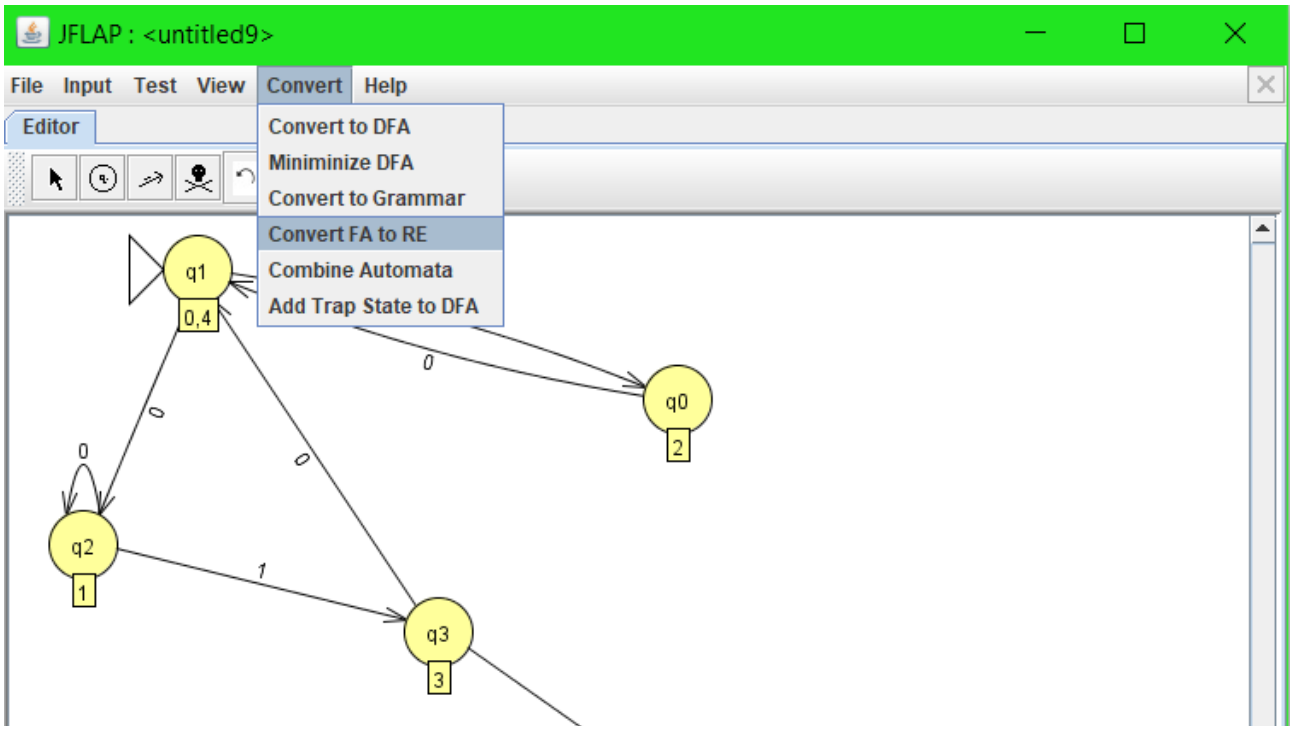


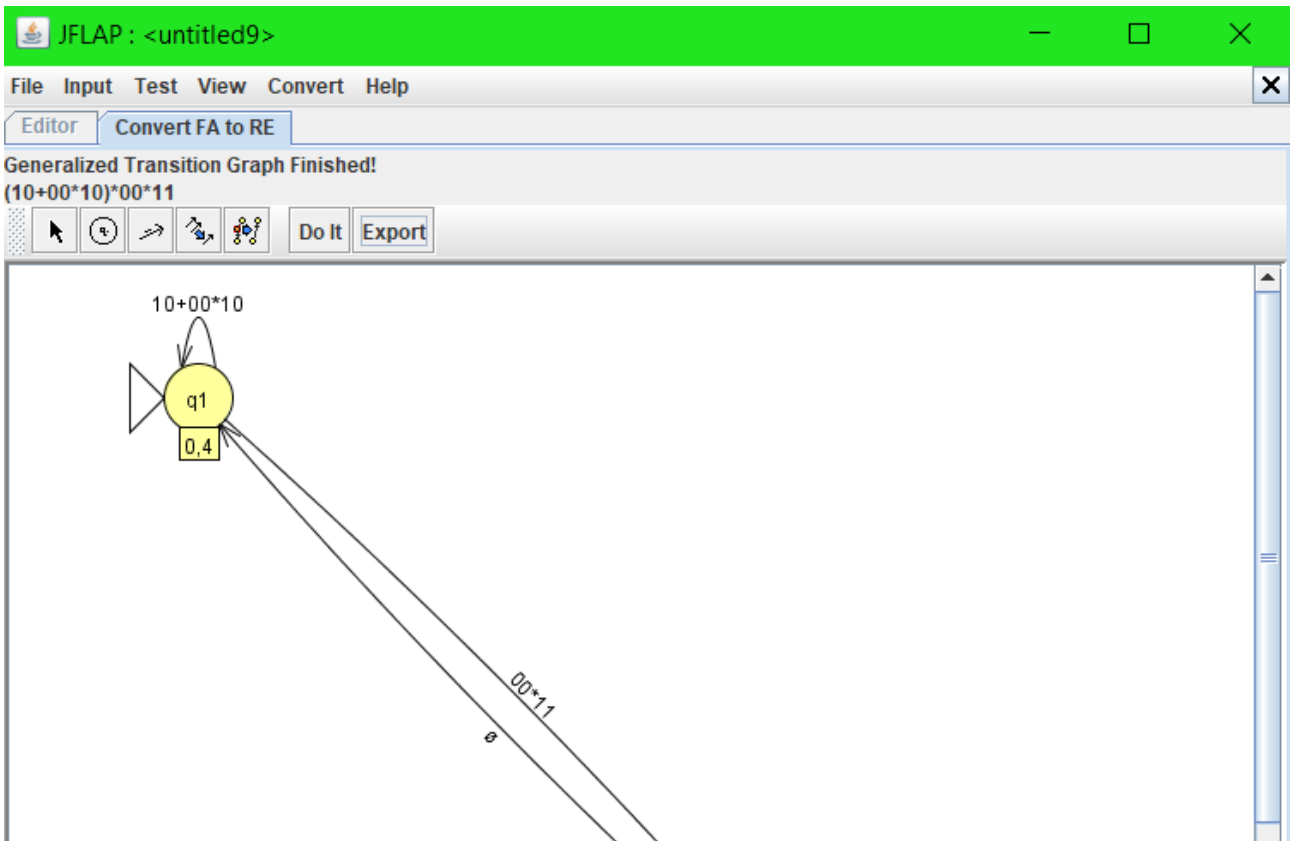
Este sería el autómata finito determinista final. No hay mejor versión que esta.



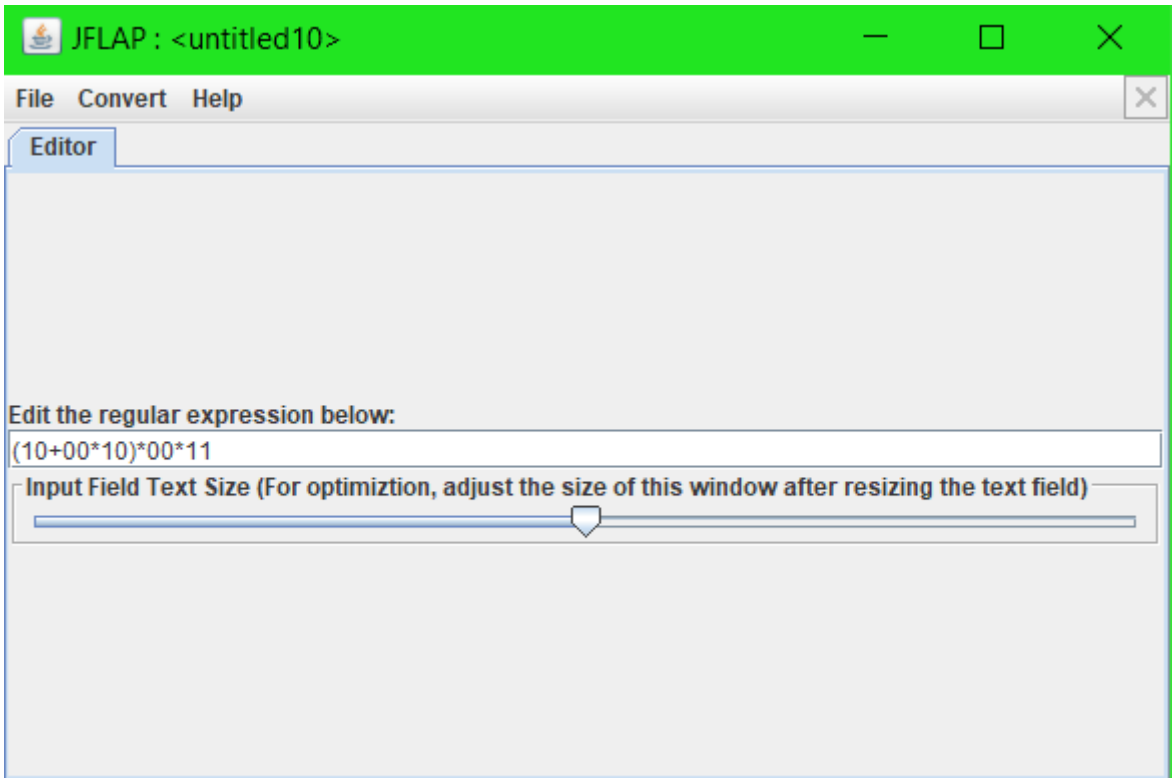
SEGUNDA PARTE Autómata → Expresión Regular

Partimos del autómata finito determinista minimizado. Lo paso a expresión regular en jflap:





Lo exportamos y nos devuelve la expresión regular equivalente a ese autómata.



Si repitiera de nuevo todo el proceso, volvería a obtener el mismo autómata. Por tanto concluimos que el autómata se representa por 2 expresiones regulares distintas, esto nos hace ver que la expresión regular no es única.

El autómata finito y la expresión regular son equivalentes.