UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informatica y Telecomunicaciones

Examen Modelos de Computación

Grupo B3

Juan Luis Torres Ramos

15 Enero 2024

Ejercicio 1

1. Obtener un modelo de calculo para $L = \{U \mid U \in \{a, b\}^{\pm}$ y $N_a(U) = N_b(U)\}$.

Para programar la gramática que genera este lenguaje, primero observamos que busca cadenas con el mismo número de b's que de a's $N_a(U) = N_b(U)$. Al aplicar el lema del bombeo, nos dice que el lenguaje no es regular por lo que intentaremos encontrar una gramática libre de contexto.

Tenemos que estudiar todos los casos posibles. Comenzamos con la producción inicial S que genera a y algo más, es decir, $S \to aX$, donde X es una variable que determinaremos. Esta producción debe garantizar que no se viole la condición $N_a(b) = N_b(b)$. Esto nos lleva a la producción $S \to aB$, con $B \to b$, esta ultima produccion simepre va a haber una b de más, asi cumplimos la condicion anterior.

De igual manera, razonamos en la dirección opuesta y obtenemos $S \to bA$ con $A \to a$. Hasta este punto, hemos definido cuatro producciones.

Ahora, estudiamos el caso de la producción $S \to aB \to abX$. Este caso nos centraremos cuando $B \to bX$,como queremos mantener la condicion de B (tener una b de más). Observamos que $B \to bS$ cumple con la condicion (S genera una a y una b para este caso). De manera análoga, obtenemos $A \to aS$ para el caso contrario.

Estudiamos el caso $B \to aX$. + Necesitamos una b adicional para cumplir la condición de B (tener una b de más), así que definimos $B \to aBB$, asegurando que la primera B compense una a y luego llamando nuevamente para tener un B adicional. De manera inversa, obtenemos $A \to bAA$.

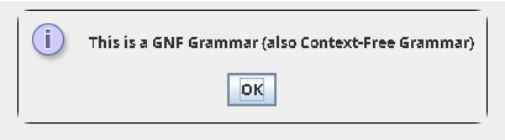
La gramática libre de contexto resultante es: $G = (V, T, P, S)$ donde $V = \{S, A, B\}$, $T = \{a, b\}$, y P es el conjunto de producciones:

$S \to aB$	$S \to bA$	$B \to bS$	$B \to aBB$
$B \to b$	$A \to a$	$A \to aS$	$A \to bAA$

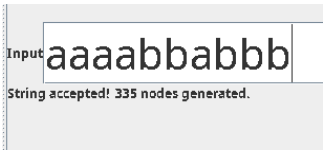
Comprobacion con JFLAP: generamos la gramatica en JFLAP y comprobamos que genera el lenguaje $L = \{U \mid U \in \{a,b\}^{\pm} \text{ y } N_a(U) = N_b(U)\}$.

LHS		
S	$\longrightarrow$	aB
S	$\longrightarrow$	bA
A	$\longrightarrow$	bAA
B	$\longrightarrow$	aBB
A	$\longrightarrow$	aS
B	$\longrightarrow$	bS
A	$\longrightarrow$	a
B	$\longrightarrow$	b

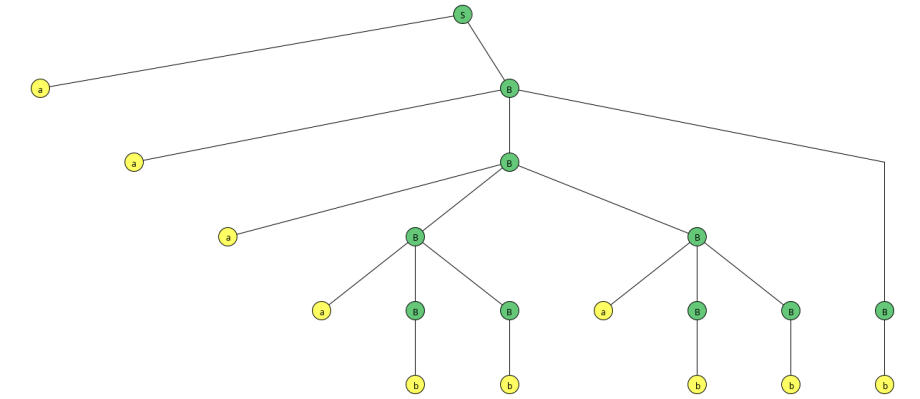
(a) La producción



(b) Comprobación gramatica libre contexto



(c) cadena *aaaabbabbb*



(d) árbol de produccion de la cadena *aaaabbabbb*

Ejercicio 2

1. **Determina si la gramática $G = (\{S, A, B\}, \{a, b, c, d\}, P, S)$ donde P es el conjunto regla producción genera un lenguaje tipo 3:**

   $S \to AB$ $\quad$ $A \to Ab$ $\quad$ $A \to a$
   $B \to cB$ $\quad$ $B \to d$

2. **obtener el ATDM, (automanta deterministico minimal) Obtener el modelo de cálculo más optimo para resolver el problema del apartado A**

Apartado 1: Es un tipico problema de optimizacion. Primero nos fijamos en que no es una gramatica regular por producciones como $B \to cB$ o $A \to Ab$ Generar un lenguaje de tipo 3 significa que sea libre de contexto. Estudio una produccion y genero una cadena de ejemplo:

$$S \to AB \to AbB \to AbbB \to abbcB \to abbccB \to abbcccB \to abbcccd$$

Si nos fijamos genera el lenguaje $L = \{ab^i c^j d : i, j \in \mathbb{N}\}$ (es decir genera al principio una a, al final una d y entre medias un numero i de b y un numero j de c, en el orden descrito). Vemos si existe una gramatica libre de contexto que genere este lenguaje, es decir, optimizamos.

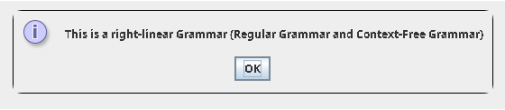
Este lenguaje se genera con la gramatica $G = (\{S, A, B\}, \{a, b, c, d\}.P, S)$ donde P es el conjunto regla producción

$S \to aB$ $\quad$ $B \to C$ $\quad$ $C \to d$
$B \to bB$ $\quad$ $C \to cC$

Comprobacion con JFLAP

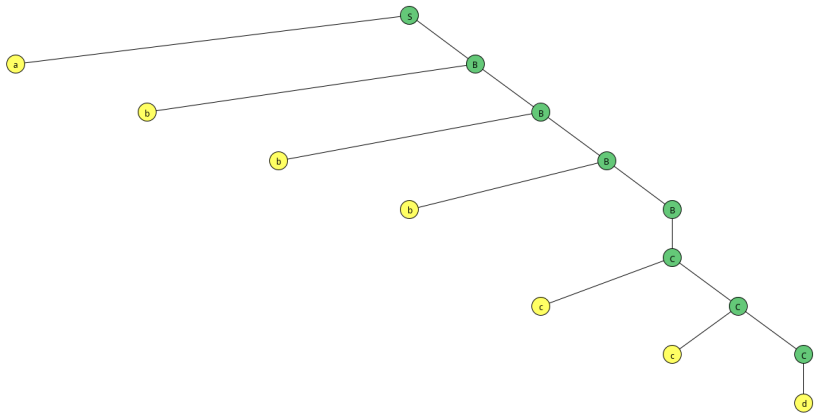
LHS		RHS
S	→	aB
B	→	bB
B	→	C
C	→	cC
C	→	d

(a) la producción



(b) gramatica libre contexto

(c) cadena *abbbccd*

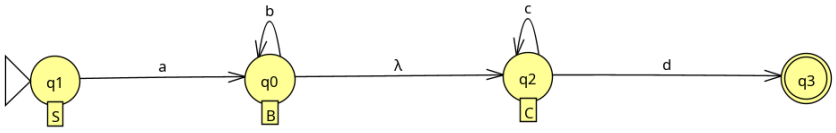


(d) árbol de produccion de la cadena *aabb*

Apartado 2: Quiero crear un autómata no determinístico con transiciones nulas. Con la produccion anterior en JFLAP he insertado la gramatica y luego la he convertido a automata finito (Convert-right linear gramar to FA)

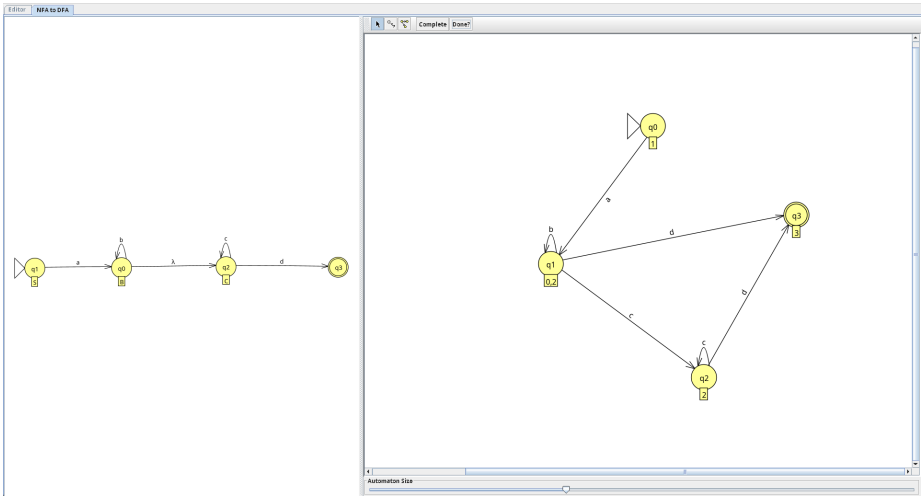
Production	Created
S→aB	☑
B→bB	☑
B→C	☑
C→cC	☑
C→d	☑

(a) Producción



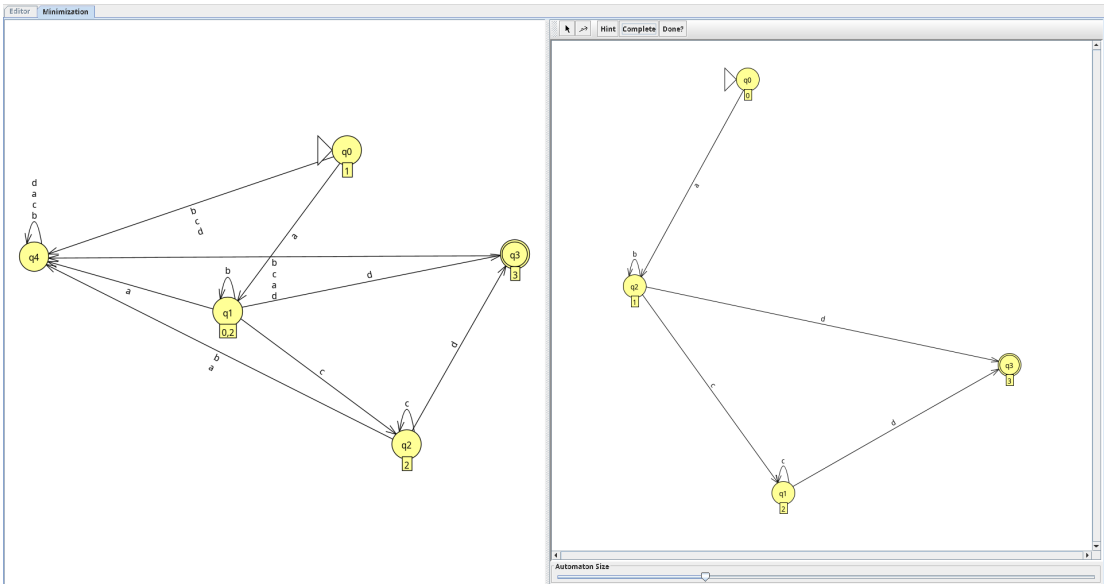
(b) Automata Finito con transiciones nulas generado

Con un FA, lo he convertido a automata finito deterministico (convert to DFA)

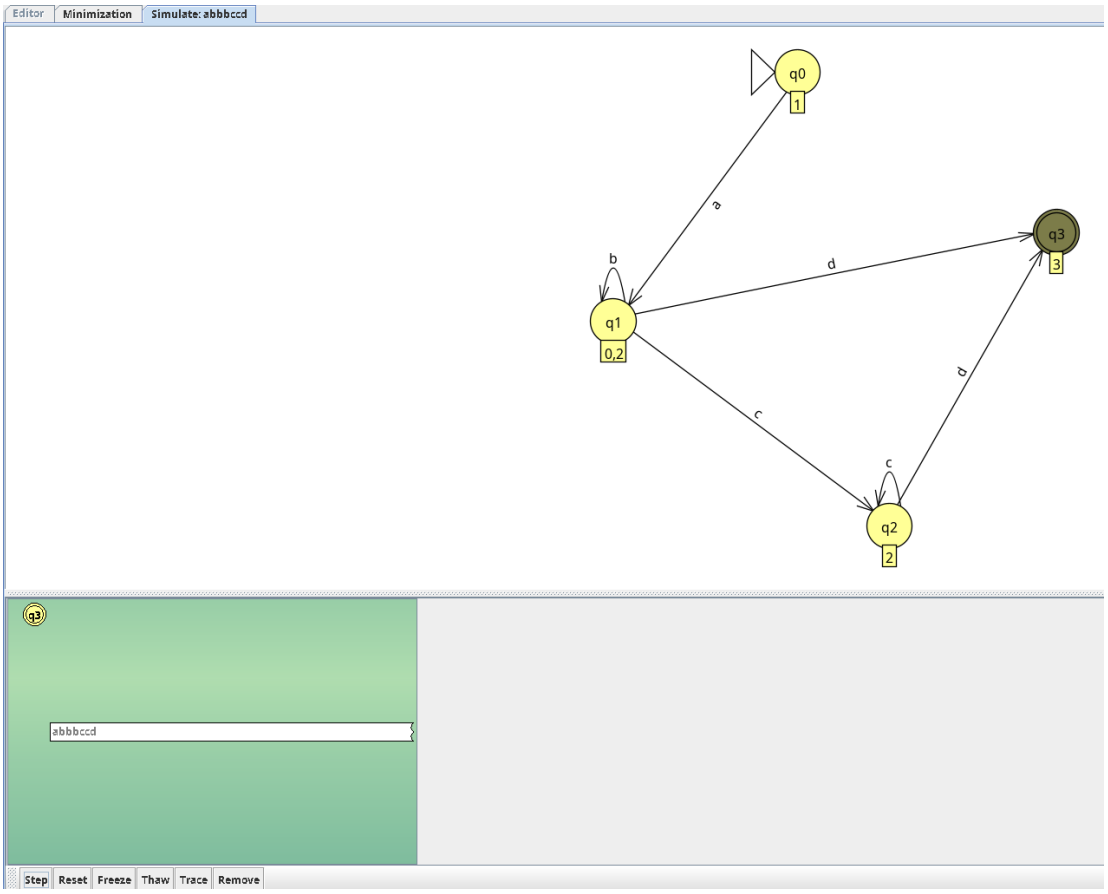


(c) Automata finito deterministico

Automata finito deterministico minimal desde el DFA elo minimozo, vemos que no se puede minimizar por que da el mismo que el anterior



(d) Automata finito deterministico minimal



(e) Ejemplo con la cadena *abbbccd*

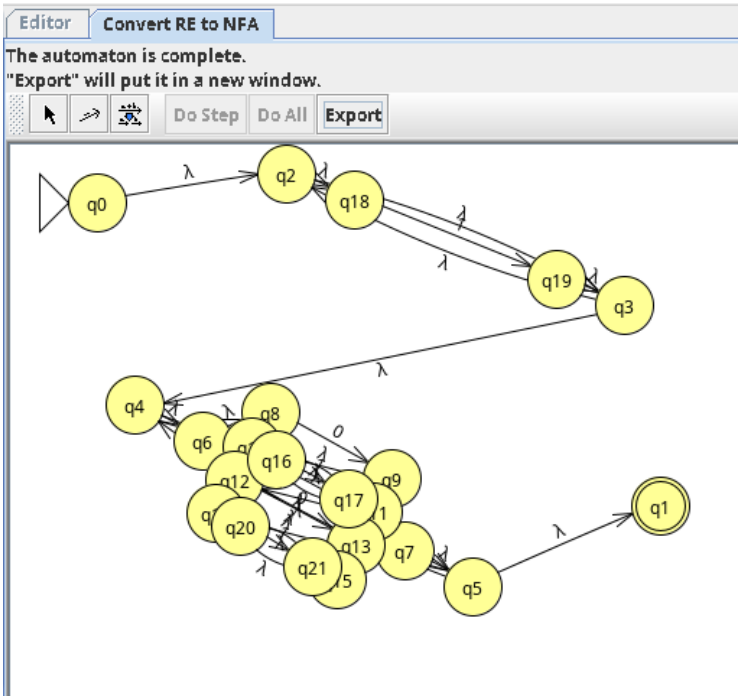
Ejercicio 3

Apartado 1: construir una expresión regular para las palabras en las que numeros de ceros es par

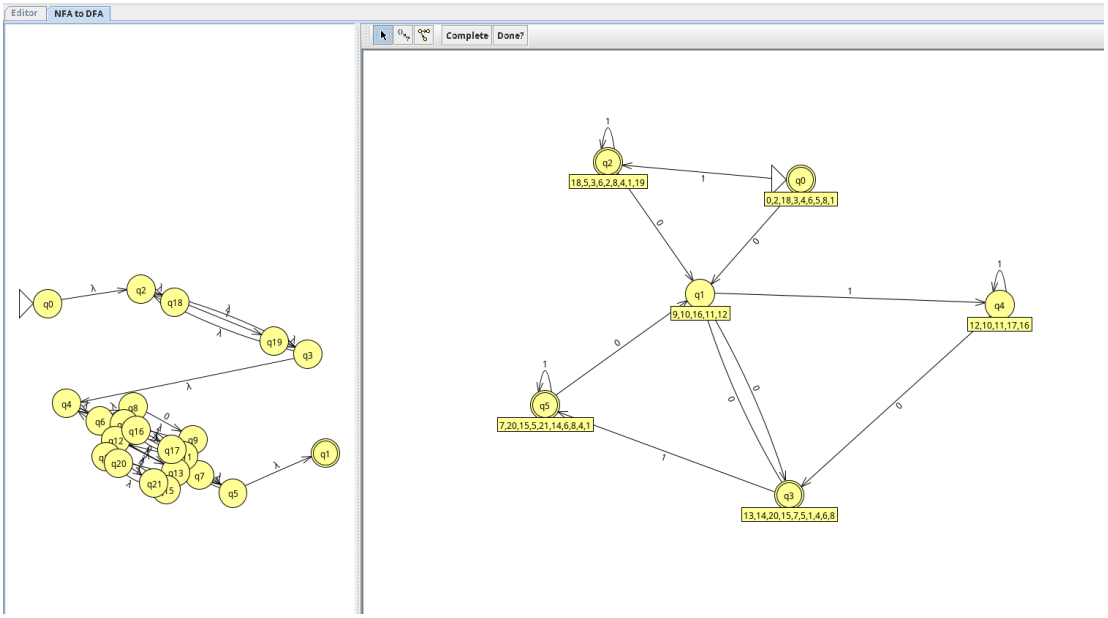
$$1 * (01 * 01*)*$$

Forzamos a de esta manera que comience por 1 y que vayan apareciendo ceros de manera par, de dos en dos . Simulamos en JFLAP, Convertimos la expresion regular a un automata finito deterministico minimal

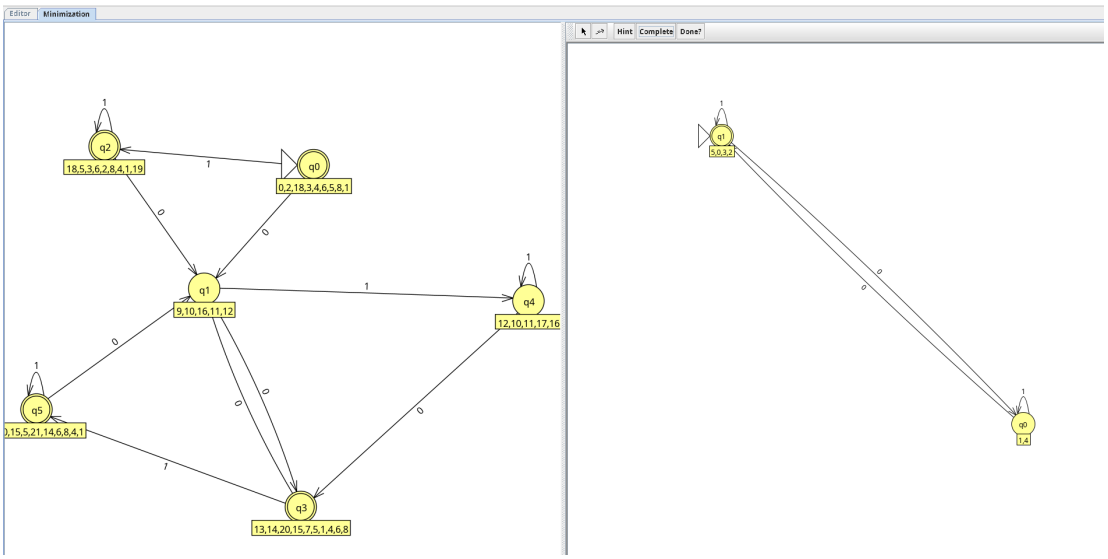
(a) Expresion regular



(b) Automata finito no deterministico



(c) Automata finito deterministico



(d) Automata finito deterministico minimal

Table Text Size

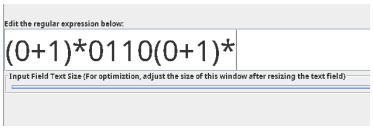
Input	Result
1111111	Accept
000010	Reject
11101110	Accept
000010000	Accept

(e) Ejemplo de ejecucion

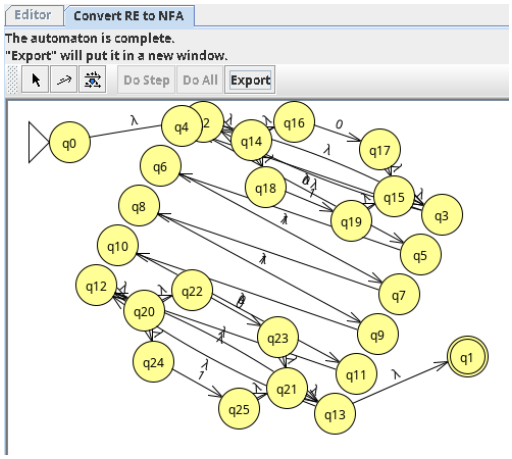
Apartado 2: Construir una expresion regular para las palabras que contengan a 01100 como subcadena

$$(0+1)*0110(0+1)*$$

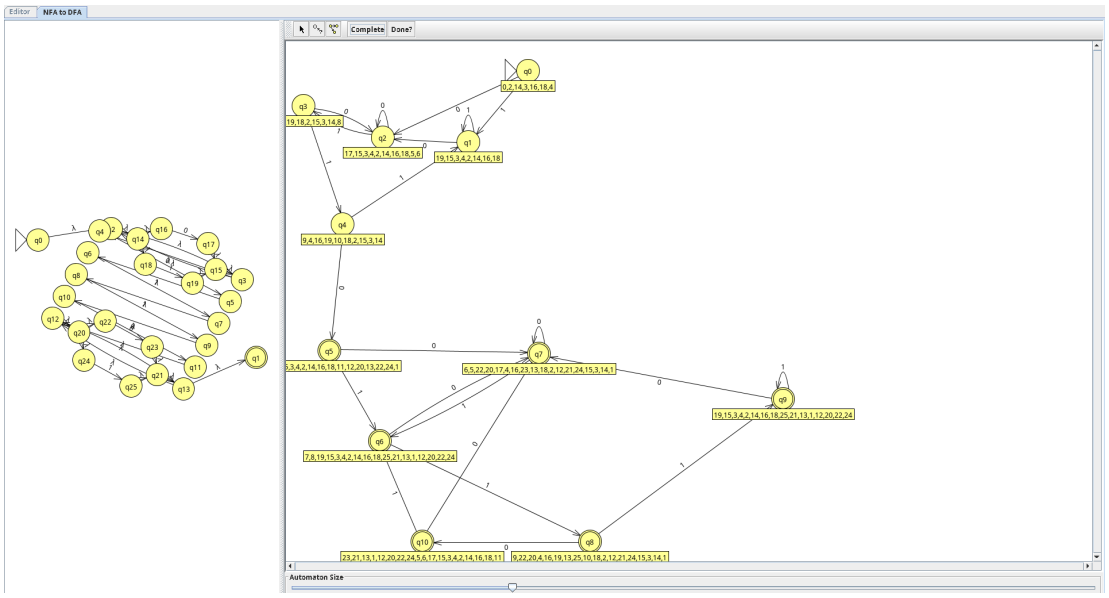
Una cadena que comience y termine por cualquier combinacion de 0 y 1 y contenga 0110 . Simulamos en JFLAP, Convertimos la expresion regular a un automata finito deterministico minimal



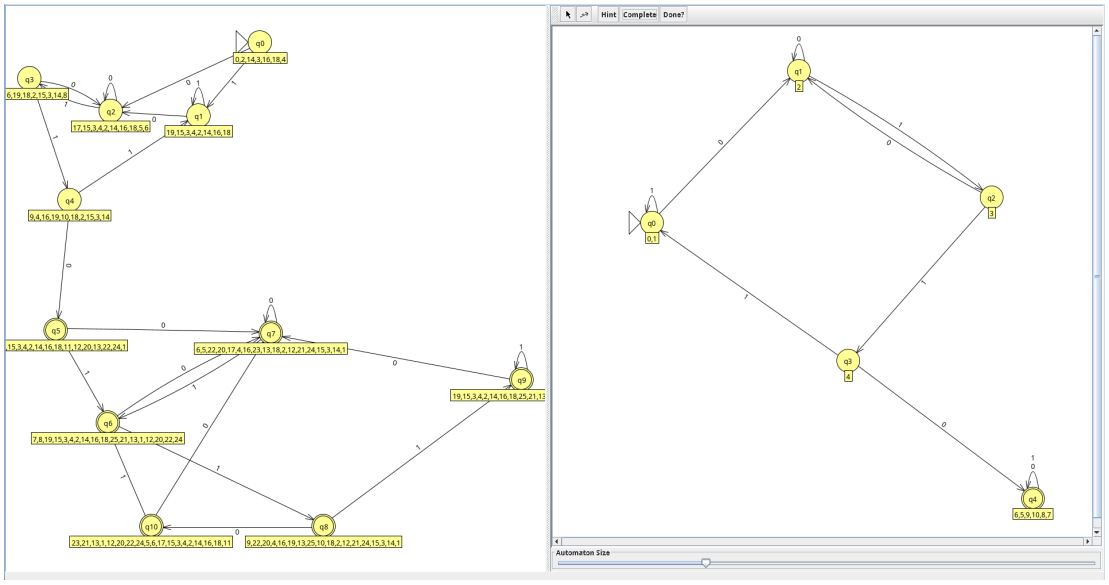
(a) Expresion regular



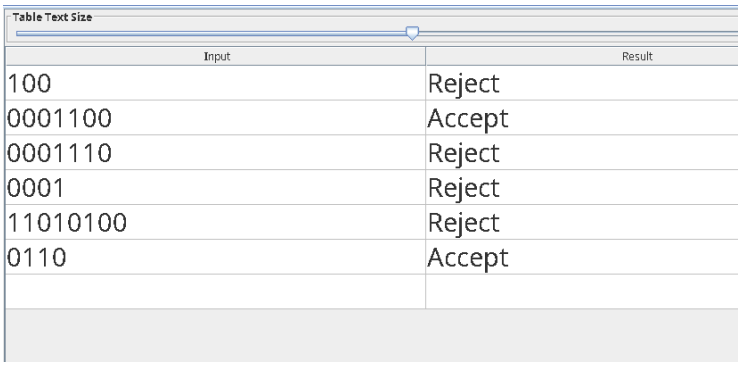
(b) Automata finito no deterministico



(c) Automata finito deterministico



(d) Automata finito deterministico minimal



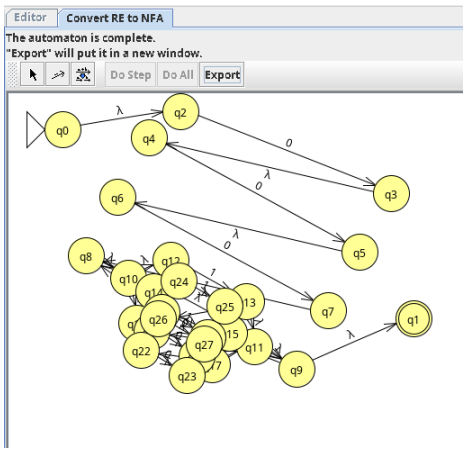
(e) ejemplo de ejecucion

Apartado 3: Construir una expresion regular para el conjunto de palabras que empiezan por 000 y tales que esta subcadena solo se encuentra al principio de la palabra

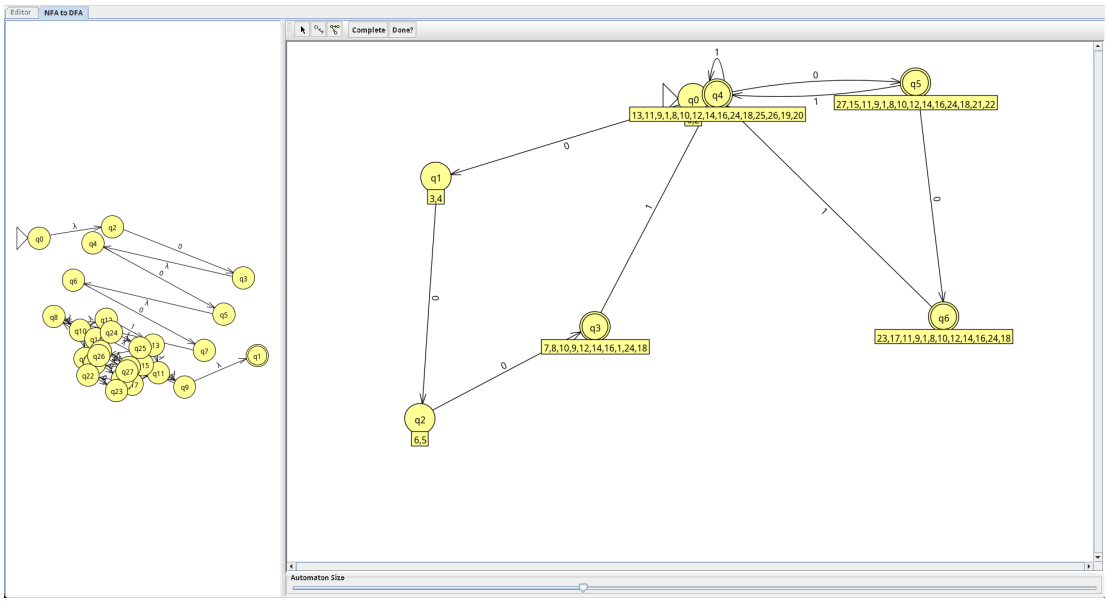
$$000(1 + 10 + 100)*$$

Comienza por 000 y luego no vuelve a aparecer 000. Simulamos en JFLAP, Convertimos la expresion regular a un automata finito deterministico minimal

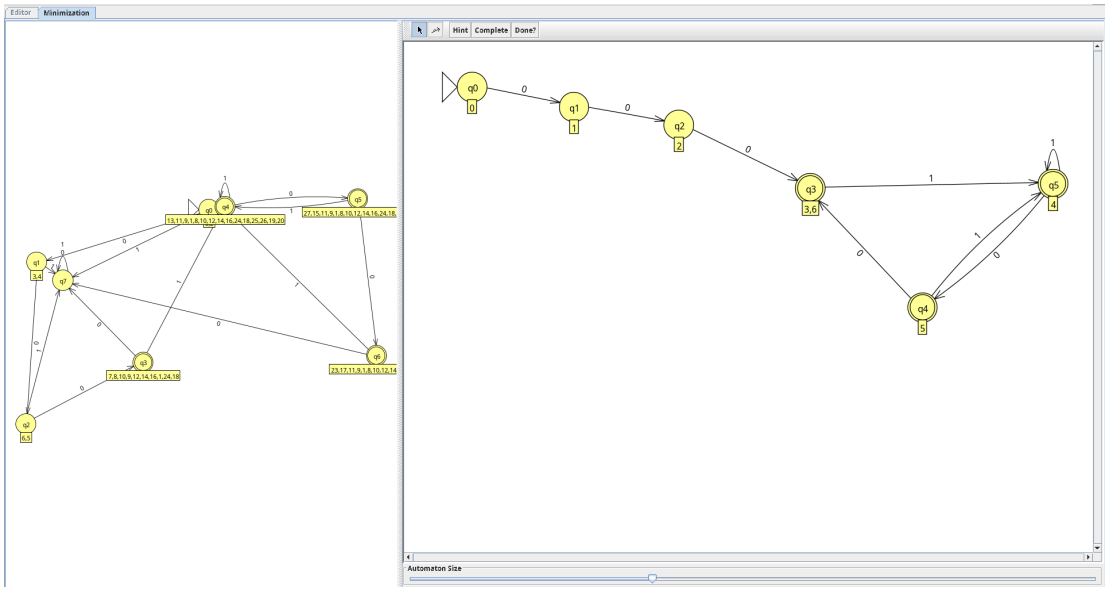
(a) Expresion regular



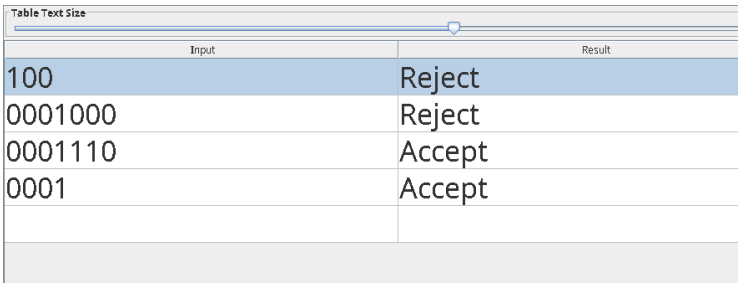
(b) Automata finito no deterministico



(c) Automata finito deterministico



(d) Automata finito deterministico minimal



(e) ejemplo de ejecucion

Ejercicio 4

1. **¿Es $L = \{U \in \{0,1\}^* / \ u = u^{-1}\}$. regular?**
2. **Encontrar un modelo de calculo para L**

El lenguaje $L = \{U \in \{0,1\}^* / \ u = u^{-1}\}$ es el conjunto de cadenas binarias que son palíndromos (secuencias de 0 y 1). Un palindromo es una cadena que se lee igual de izquierda a derecha. Para saber si es regular o no aplico el Lema de Bombeo.

Definimos el Lema de Bombeo: Sea L un conjunto regular. Entonces, existe un $n \in \mathbb{N}$ tal que para todo $z \in L$ con $|z| \geq n$, se puede expresar como $z = uvw$ donde:

1. $|uv| \leq n$
2. $|v| \geq 1$
3. Para todo $i \in \mathbb{N}$, $uv^i w \in L$

Además, el número de estados de cualquier autómata que acepta el lenguaje L es al menos n.

La idea básica del lema de bombeo es que para cualquier cadena lo suficientemente larga, puedes "bombear" o repetir una parte de la cadena y aún obtener una cadena en el lenguaje. Este lema siempre es verdadero para lenguajes regulares. Si no es verdadero para un lenguaje, entonces ese lenguaje no es regular. Estudiamos el caso del contraejemplo.

Contraejemplo: L es regular y satisface el lema de bombeo.

$$\exists n \in \mathbb{N}, \quad \forall Z \in L, \quad |Z| \geq n, \quad Z = 0^n 1^n 0^n = uvw$$

Aplicamos $|uv| \leq n$:

$$uv = 0^k, \quad v = 0^l, \quad w = 0^{n-k-l} 1^n 0^n$$

$|v| \geq 1$ implica que $v = 0^l$ y $l \geq 1$.

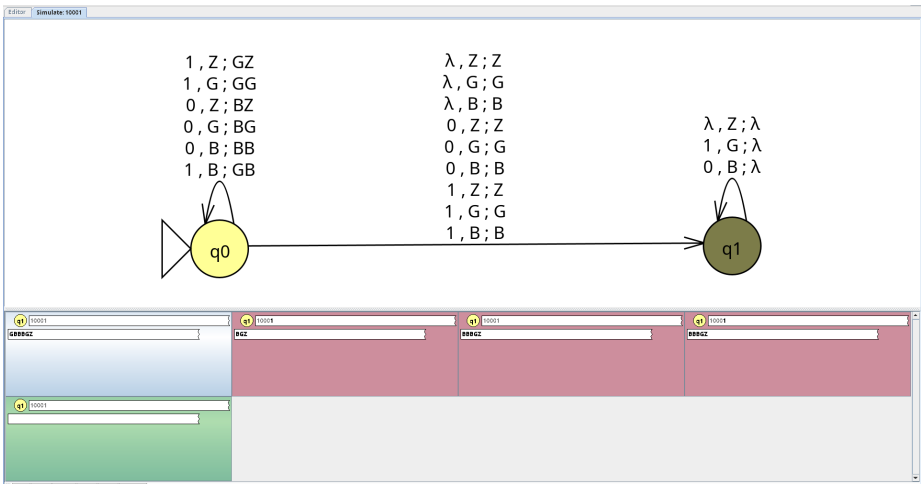
Para $i \geq 2$, verificamos $uv^i w \in L$:

$$i = 2 \quad uv^2 w = 0^k 0^{2l} 0^{n-k-l} 1^n 0^n = \quad 0^{n+l} 1^n 0^n \notin L$$

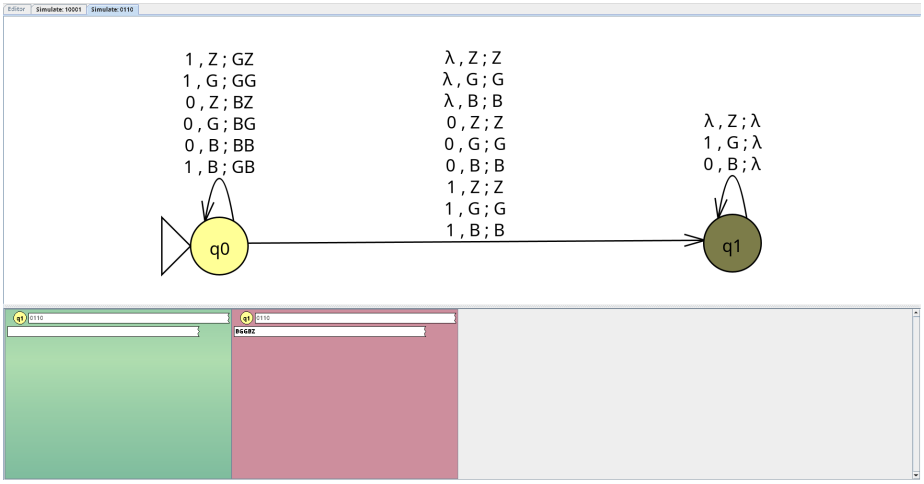
Hemos alcanzado una contradicción con el lema de bombeo, por lo que L no es regular.

Apartado 2: Obtener un modelo de calculo para L

El modelo es no deterministico, lo que significa que no diferencia cuando cambia entre estados. Simulo todos los posibles casos. $q0$ se encarga de la parte de la derecha del palindromo y $q1$ de la parte izquierda Cuando el numero es par el cambio se hace con el 0 o 1. Cuando el numero es impar el cambio se haria con λ



(a) caso impar 10001



(b) caso par 0110