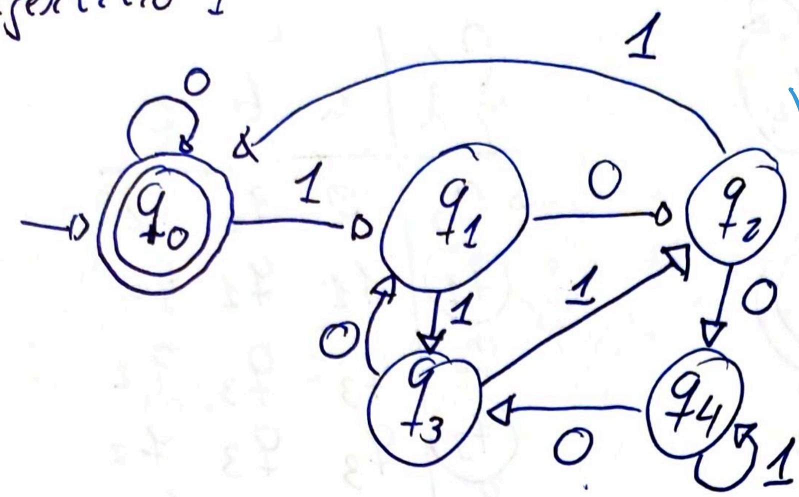
9

Muy buena práctica. Una pena que el 1 no esté completo.


**Manuel Zafra Mota**
**José Teodosio Lorente Vallecillos**

FALTA GRAMÁTICA Y REGEX :(

Ejercicio 1




| $\delta/\lambda$ | 0 | 1 |
|---|---|---|
| $\to (q_0)$ | $q_0$ | $q_1$ |
| $q_1$ | $q_3$ | $q_2$ |
| $q_2$ | $q_4$ | $q_0$ |
| $q_3$ | $q_1$ | $q_2$ |
| $q_4$ | $q_3$ | $q_4$ |

1'5/2'5

La idea es la siguiente: el autómata tiene que saber en qué miembro de $Z_5$ está, si ha recibido un número i, y ahora viene un 0, va a tener $2i \pmod 5$, y si recibe 1 es $2i+1 \pmod 5$.

Los estados $q_0, q_1, q_2, q_3$ y $q_4$ corresponden a las entradas 0, 1, 2, 3 y 4 mod 5, respectivamente.

Suponemos que hasta el momento tenemos un número multiplo de 5, portanto, estamos en el estado $q_0$.

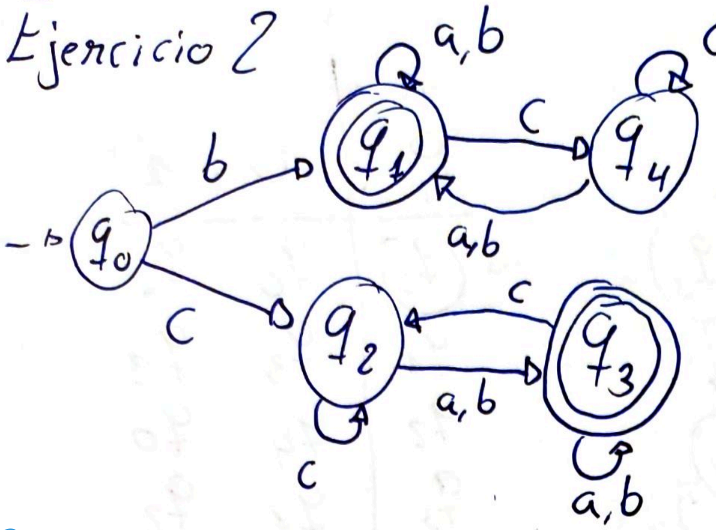
Un 0 multiplica el numero actual por 2, de modo que todavia es un múltiplo de 5, y por tanto estamos en el estado $q_0$ (estado final).

Un 1 multiplica por 2 y añade 1, 1 mod 5, portanto estamos en el estado $q_1$.

De la misma manera para los demas estados $q_1, q_2, q_3$ y $q_4$; y sus respectivos $q_i$ i mod 5.

Ejercicio 2

2'5




| $\delta/\lambda$ | a | b | c |
|---|---|---|---|
| $\rightarrow q_0$ | $\emptyset$ | $q_1$ | $q_2$ |
| $q_1$ | $q_1$ | $q_1$ | $q_4$ |
| $q_2$ | $q_3$ | $q_3$ | $q_2$ |
| $q_3$ | $q_3$ | $q_3$ | $q_2$ |
| $q_4$ | $q_1$ | $q_1$ | $q_4$ |

Se podría hacer con menos estados. Pero bien

El funcionamiento es el siguiente:

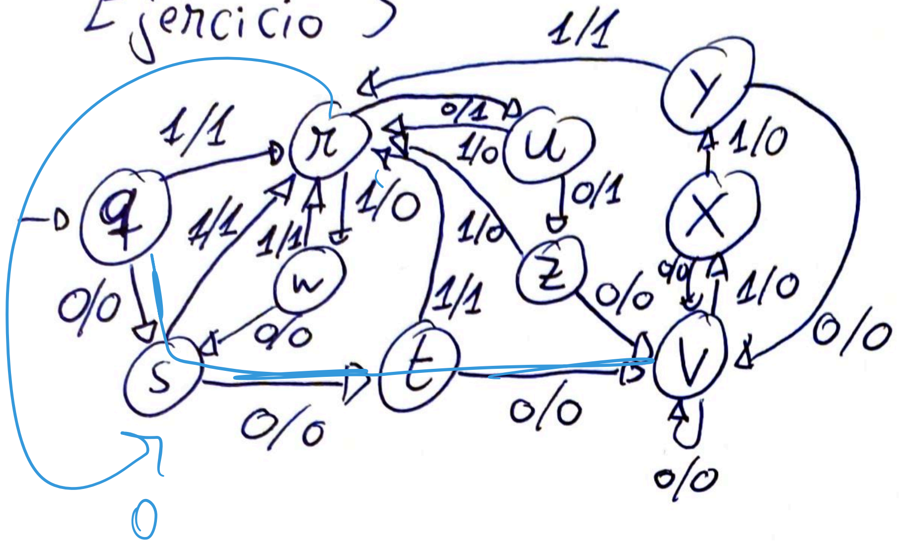
Tenemos un estado inicial, $q_0$, del cual podemos partir con b ó c, ya que el simbolo inicial no puedo ser a.

En caso de empezar con b, se sube al estado $q_1$ y el objetivo es que a partir de ahí se pueda crear cualquier palabra salvo las que cuyo simbolo final no es c, de ahí que cuando se usa la c estándo en $q_1$ vas a $q_4$, en donde puedes volver a poner todas las c necesarias, pero para volver al estádo final $q_1$ para finalizar es necesario una a ó b.

Y en caso de empeza con c, se baja al estado $q_2$ y el objetivo es el mismo, pero en este caso al no poder ser c el simbolo final el estado final es $q_3$ en vez de $q_2$.

# Ejercicio 3

2'5



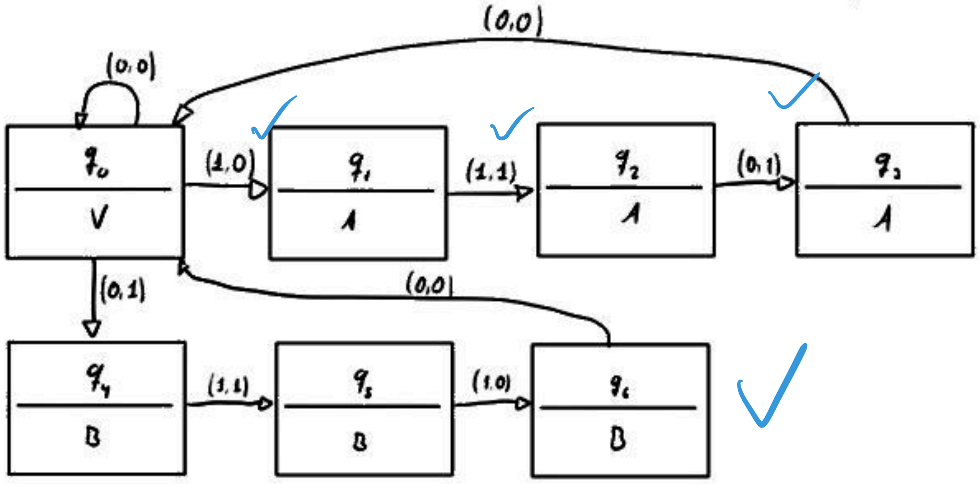

| $\delta/\lambda$ | 0 | 1 |
|---|---|---|
| $\to$ q | s/0 | r/1 |
| r | u/1 | w/0 |
| s | t/0 | r/1 |
| t | v/0 | r/1 |
| u | z/1 | r/0 |
| v | v/0 | x/0 |
| w | s/0 | r/1 |
| x | v/0 | y/0 |
| y | v/0 | r/1 |
| z | v/0 | r/0 |

Para crear la máquina, simplemente es ver si la suma binaria sin acarreo ó xor hasta el momento es 0 ó 1, y en el caso de ser 0 y el bit recibido actual es 0 la salida es 0 y se empieza a contar sino había empezado ya ha contar la secuencia 000, para desactivar la salida, y se queda ala espera de recibir como entrada la secuencia 111 para volver al comportamiento previo, en el otro caso en el que el bit recibido fuese 1, la salida es 1; si hasta el momento el xor es 1, y el bit recibido es 0 la salida es 1 y se empieza a contabilizar la secuencia 000, en el caso de ser 1 el bit recibido, la salida es 0.

Todo esto teniendo en cuenta que la salida no se encontrase desactivada previamente, ya que, en ese caso nada serviría, las salidas son 0 y a la espera de la secuencia 111.

4.) Para la implementación de este sistema, utilizaré una máquina de Moore, en la que hay dos sensores que detectan cuando pasa un coche por ellos: $e_1$ y $e_2$. Dependiendo del orden en el que los sensores se activan, podemos saber si los vehículos van en la dirección correcta. Si se activa primero $e_1$ y acto seguido $e_2$, entendemos que un coche se desplaza en el sentido correcto.

El alfabeto de entrada está formado por los pares $(i,j)$, $i,j = 0,1$; donde "$i$" indica si $e_1$ detecta un vehículo, y "$j$" lo mismo para "$e_2$". El alfabeto de salida está formado por el estado de la carretera en relación al vehículo, estos pueden ser: V(vacío), A(dirección correcta), B(dirección inversa).




Se podrían añadir estados de error. Pero está bien.

2'5