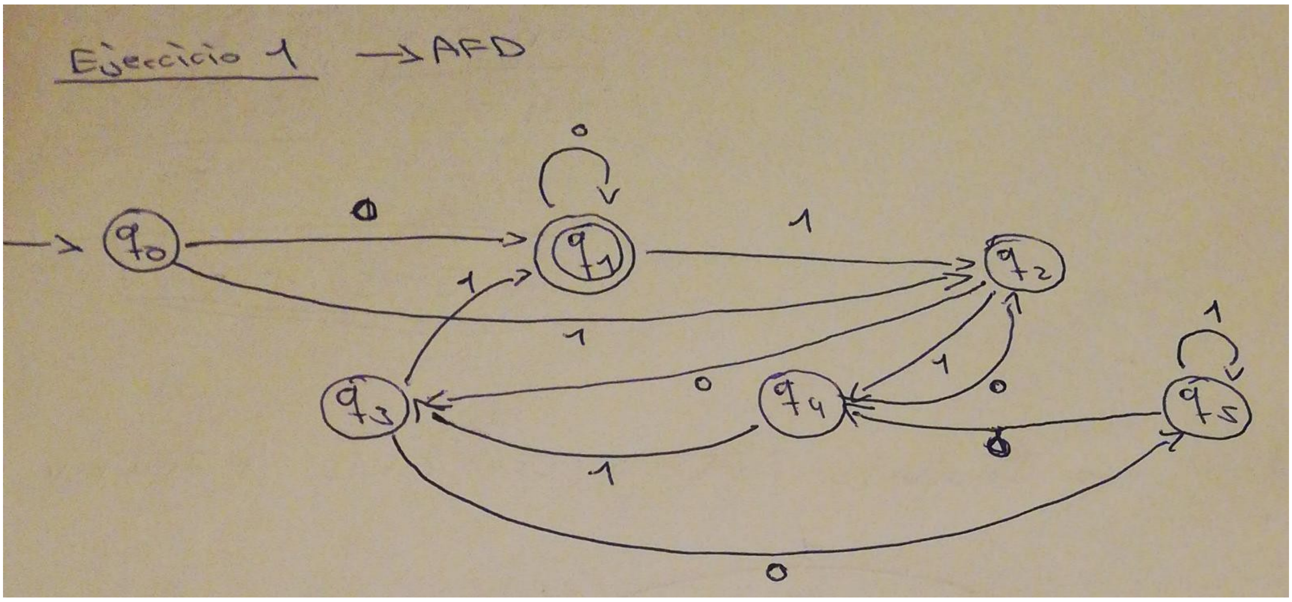
Práctica 3 – Modelos de Computación

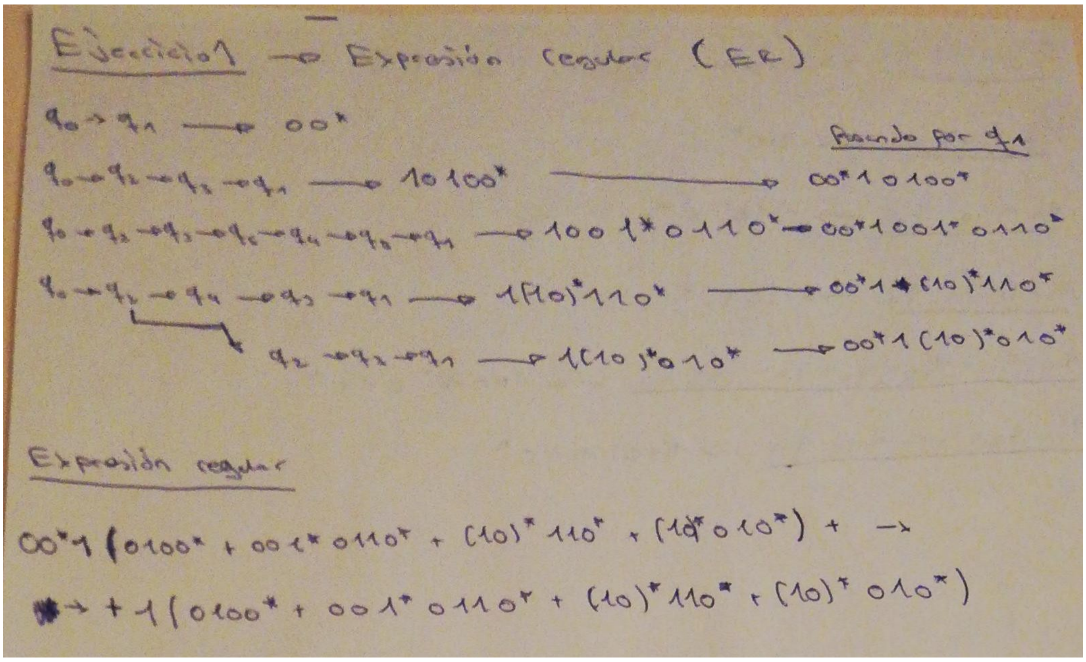
Ejercicio 1 → Calcular un autómata finito determinista que acepte el lenguaje de las palabras formadas por 0's y 1's que representan los números en binario divisibles por 5. Calcula una gramática regular por la izquierda que genere el mismo lenguaje. Calcula una expresión regular que describa este lenguaje.

-Autómata finito determinista (AFD).



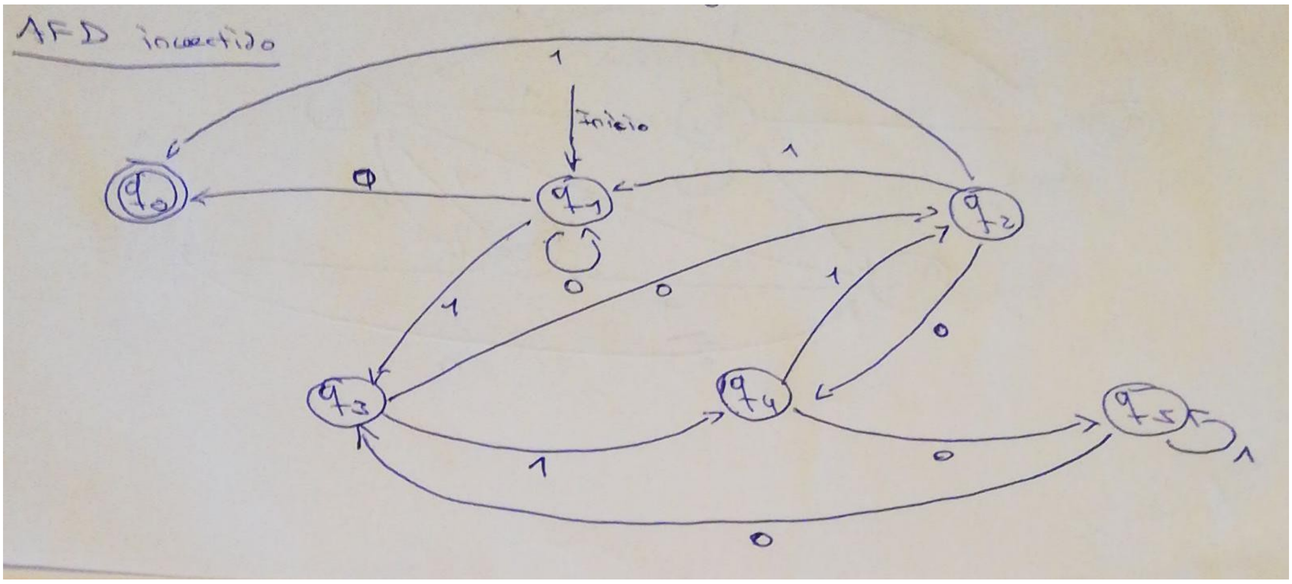
Cada estado representa el "contador" actual módulo 5; esto es, si hemos procesado m dígitos, el estado actual indica m(mod5). En binario, m0 es igual que 2m, por lo que pasamos al estado que represente 2m(mod5); y m1 es igual que 2m+1 por lo que pasamos al estado que represente (2m+1)(mod5). El único estado final es 0 ya que representa 0(mod5).

Expresión regular (ER).

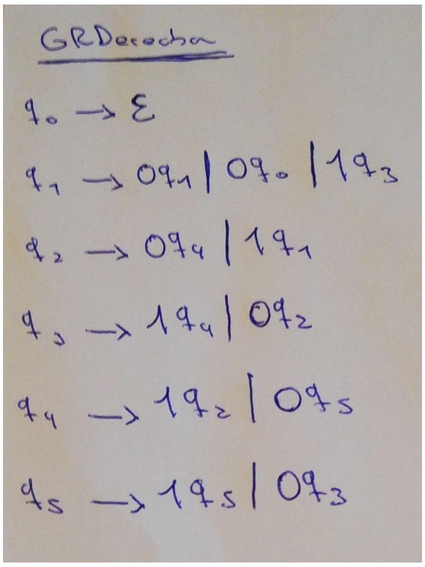


-Gramática regular por la izquierda.

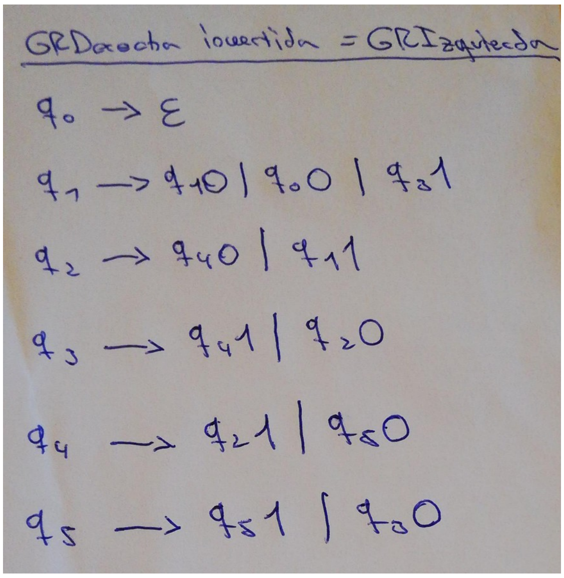
El primer paso para hallar una gramática regular por la izquierda correspondiente es invertir el AFD inicial, obteniendo así el siguiente AFD:



Tras obtener el AFD invertido, hallamos una gramática regular <u>por la derecha</u> de dicho autómata y posteriormente, la invertimos para hallar la gramática regular por la izquierda deseada, obteniendo el siguiente resultado (Imagen derecha → resultado final):

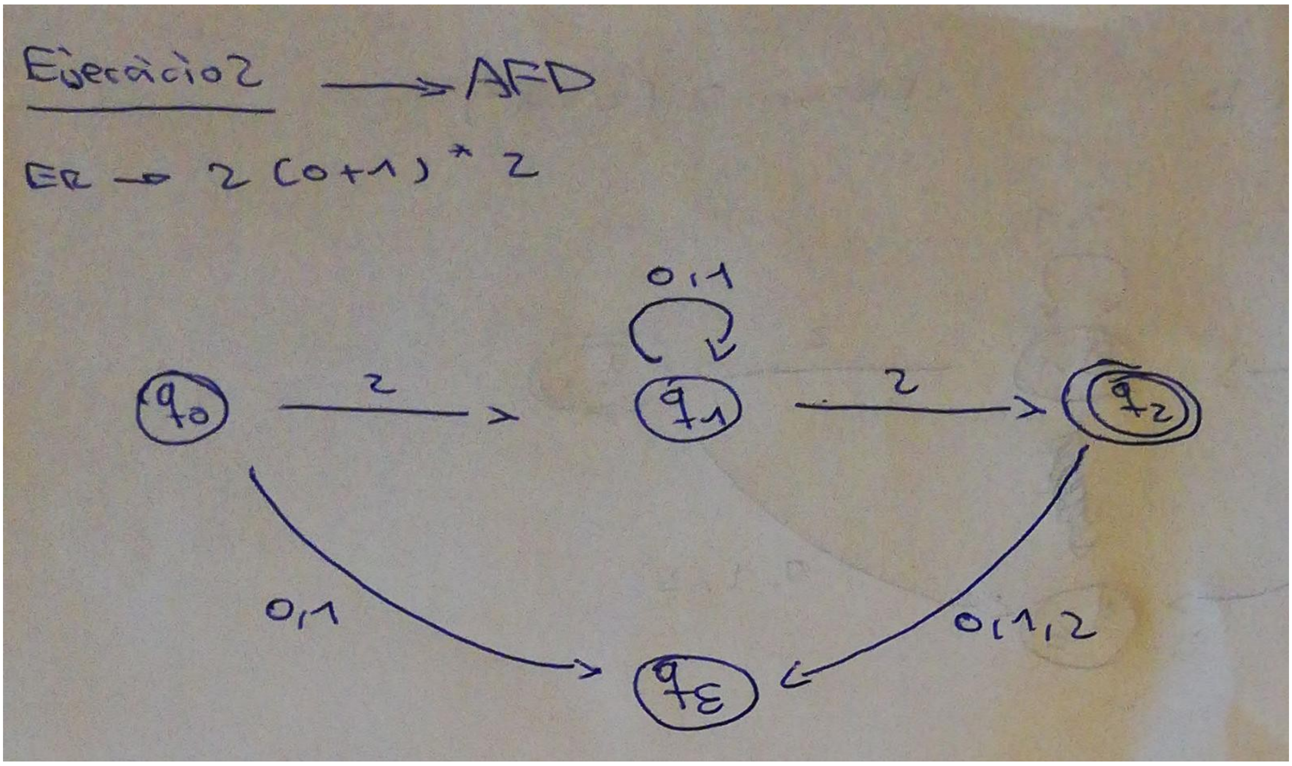


Inversión →



Ejercicio 2 → Calcular un autómata finito determinista que acepte el lenguaje de las palabras formadas por 0's, 1's y 2's que empiezan y terminan por 2 pero no contienen ningún otro 2 entre medias.

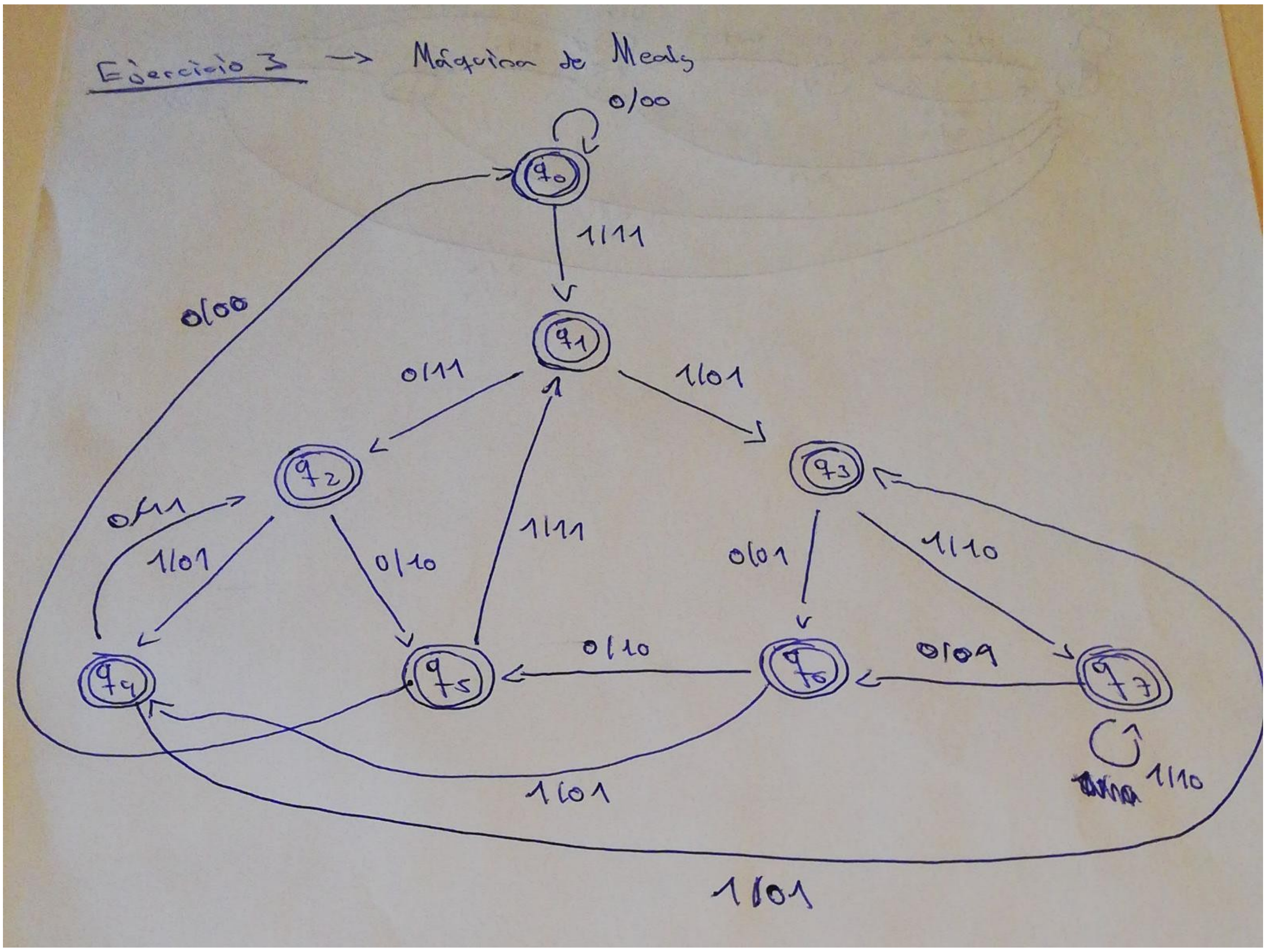
-Autómata finito determinista (AFD).



En este autómata, se parte desde el estado inicial con un 2, llegando al estado q1, donde podemos generar tantos 0’s y 1`s como deseemos en cualquier orden hasta que se introduzca un 2, lo que nos lleva al estado final donde terminamos. En caso de partir con 0 ó 1 del estado inicial, llegaremos a un estado de error. Ídem para el estado final con 0, 1 ó 2.

Ejercicio 3 → Calcula una máquina de Mealy que codifique secuencias de 0's y 1's de la siguiente manera:
-Para cada bit recibido devuelve dos bits.
-El primero es la suma (binaria) del bit recibido y de los dos anteriores.
-El segundo es la suma (binaria) del bit recibido y el anterior.
Suponemos que los dos bits recibidos antes que el primero son ambos cero. Por ejemplo, la secuencia 0101011 tiene como salida 00 11 11 01 11 01 00.

-Máquina de Mealy.



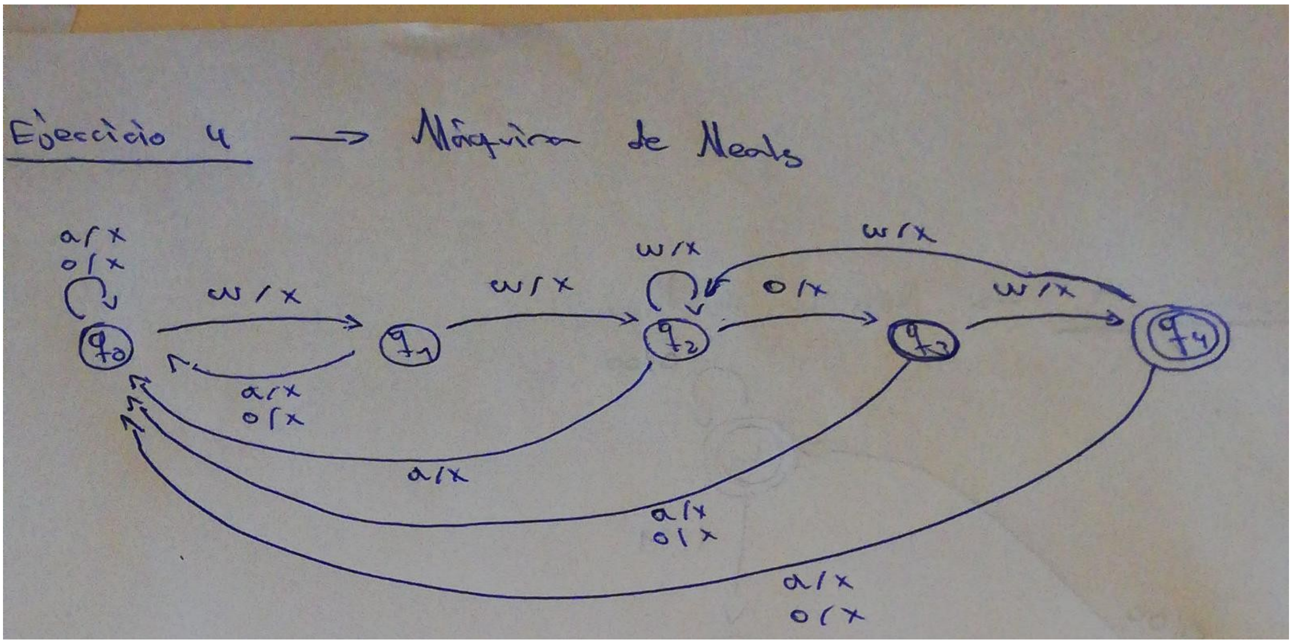
Casos posibles con la forma (segundo bit anterior, bit anterior, bit actual)=codificación

-q0 → (0, 0, 0) = 00
-q1 → (0, 0, 1) = 11
-q2 → (0, 1, 0) = 11
-q3 → (0, 1, 1) = 01
-q5 → (1, 0, 0) = 10
-q4 → (1, 0, 1) = 01
-q6 → (1, 1, 0) = 01
-q7 → (1, 1, 1) = 10

Cada estado corresponde a una combinación y pasan a otro estado tras añadir 1 ó 0. Cada arco contiene bit actual/codificación.

Ejercicio 4 → Diseñar una máquina de Mealy o de Moore que, dada una cadena de entrada usando el alfabeto A = {a, w, o}, encienda un led verde (salida V) cada vez que se detecte la cadena wwow en la entrada, apagándolo cuando lea cualquier otro ssímbolo después de esta cadena (representamos el led apagado con la salida X). El autómata tiene que encender el led verde tantas veces como aparezca en la secuencia wwow en la entrada, incluso cuando dos de estas secuencias puedan estar solapadas.

-Máquina de Mealy.



He interpretado que dos cadenas wwow están solapadas si la segunda (y las que puedan seguirle en caso de haberlas) empieza por la última w de la cadena anterior, esto es así porque si no no se cumpliría led=V tantas veces como cadenas haya, ya que después de una de las w habría o, que es un símbolo distinto de w.

Ejemplo → *wwo***w**wow son 2 cadenas wwow solapadas donde la segunda está subrayada y la primera en cursiva. Comparten como se observa la w oscura. De esta forma, led=V dos veces. Por su parte, ww**wwoow**w no sería válido ya que led=V una única vez.

Con respecto a la máquina, en cuanto se alcanza q3 y se lee w, obtenemos la cadena esperada wwow.

Nombre: Arturo Alonso Carbonero
DNI: 75936665-A
Grupo: 3ºA - A1