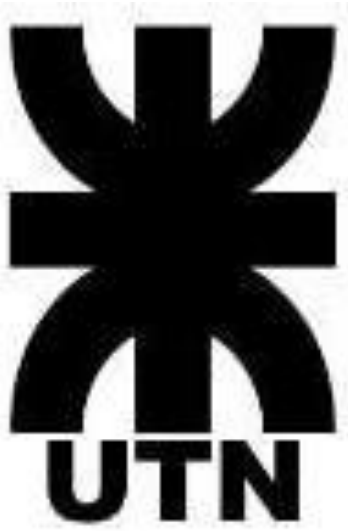


Universidad Tecnológica Nacional
Facultad Regional Villa María

Ingeniería en Sistemas de la Información

Sintaxis y Semántica del Lenguaje

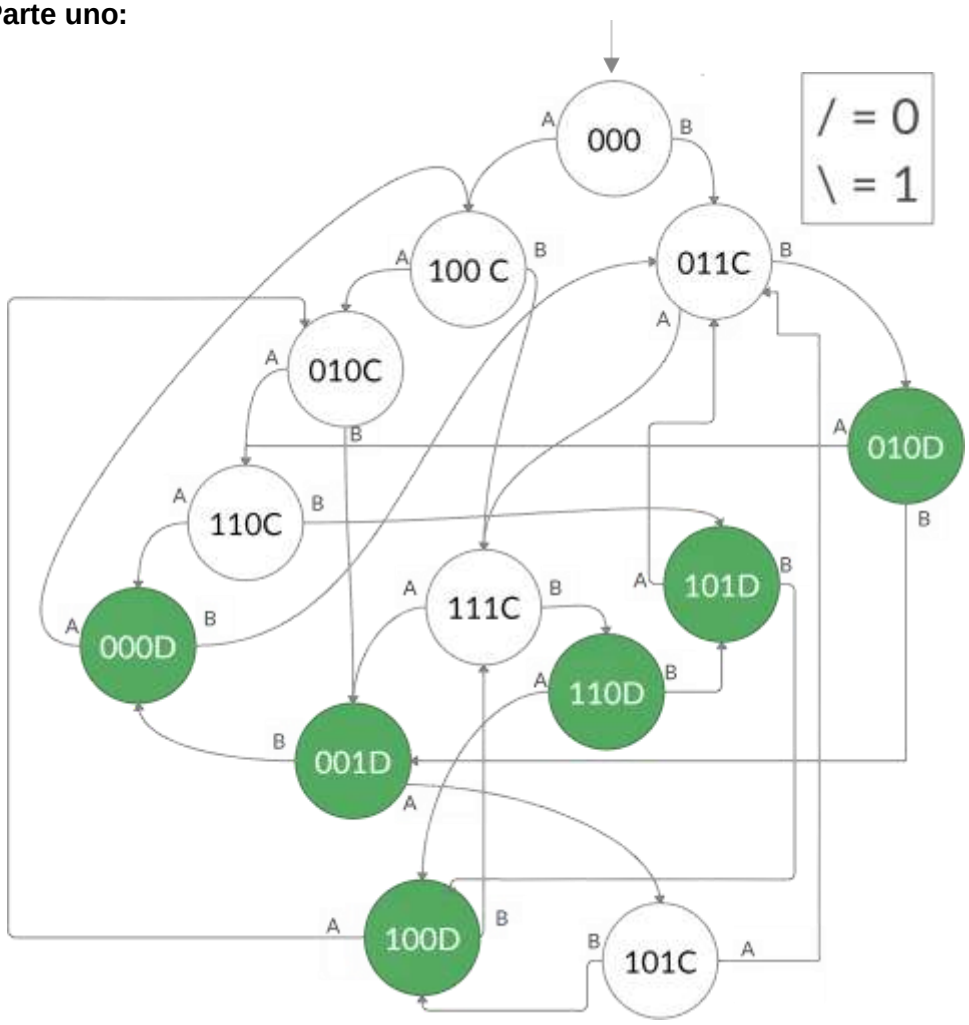
Trabajo Práctico N°2

GRUPO H

Alumnos:

- Arias Matías [matiasarias384@gmail.com][13673]
- Márquez Juan Cruz [marquezjuanchy@hotmail.com][13359]
- Muzillo Tomás [tomimuzzillo@gmail.com][13765]
- Zoy Eder [ederzoy6@gmail.com][13620]

1- A) **Parte uno:**



Parte dos:

CUADRO DE TRANSICIONES

Función de transición

Input /Estado	A	A	A	B	B	B
000	100 C	010 C	110 C	011 C	010 D	001 D
001	101 C	011 C	111 C	000 D	011 C	010 D
010	110 C	000 D	100 C	001 D	000 D	011 C
011	111 C	001 D	101 C	010 D	001 D	000 D
100	010 C	110 C	000 D	111 C	110 D	101 D
101	011 C	111 C	001 D	100 D	111 C	110 D
110	000 D	100 C	010 C	101 D	100 D	111 C
111	001 D	101 C	011 C	110 D	101 D	100 D

```
from automata.fa.dfa import DFA
# DFA which matches all binary strings ending in an odd number of '1's
dfa = DFA(
    states={'000C', '001C', '010C', '011C', '100C', '101C', '110C', '111C', '000D', '001D', '010D', '011D', '100D',
            '101D', '110D', '111D'},
    input_symbols={'a', 'b'},
    transitions={
        '000C': {'a': '100C', 'b': '011C'},
        '001C': {'a': '101C', 'b': '000D'},
        '010C': {'a': '110C', 'b': '001D'},
        '011C': {'a': '111C', 'b': '010D'},
        '100C': {'a': '010C', 'b': '111C'},
        '101C': {'a': '011C', 'b': '100D'},
        '110C': {'a': '000D', 'b': '101D'},
        '111C': {'a': '001D', 'b': '110D'},
        '000D': {'a': '100C', 'b': '011C'},
        '001D': {'a': '101C', 'b': '000D'},
        '010D': {'a': '110C', 'b': '001D'},
        '011D': {'a': '111C', 'b': '010D'},
        '100D': {'a': '010C', 'b': '111C'},
        '101D': {'a': '011C', 'b': '100D'},
        '110D': {'a': '000D', 'b': '101D'},
        '111D': {'a': '001D', 'b': '110D'},
    },
    initial_state='000C',
    final_states={'000D', '001D', '010D', '011D',
                  '100D', '101D', '110D', '111D'}

)

cadena = input("Ingresar cadenas de a y b: ")

if dfa.accepts_input(cadena):
    print('accepted')
else:
    print('rejected')


```

Verificación de cadenas:

```
Ingresar cadenas de a y b: bbb
accepted

Process finished with exit code 0
```

```
Ingresar cadenas de a y b: aaaaa
rejected

Process finished with exit code 0
```

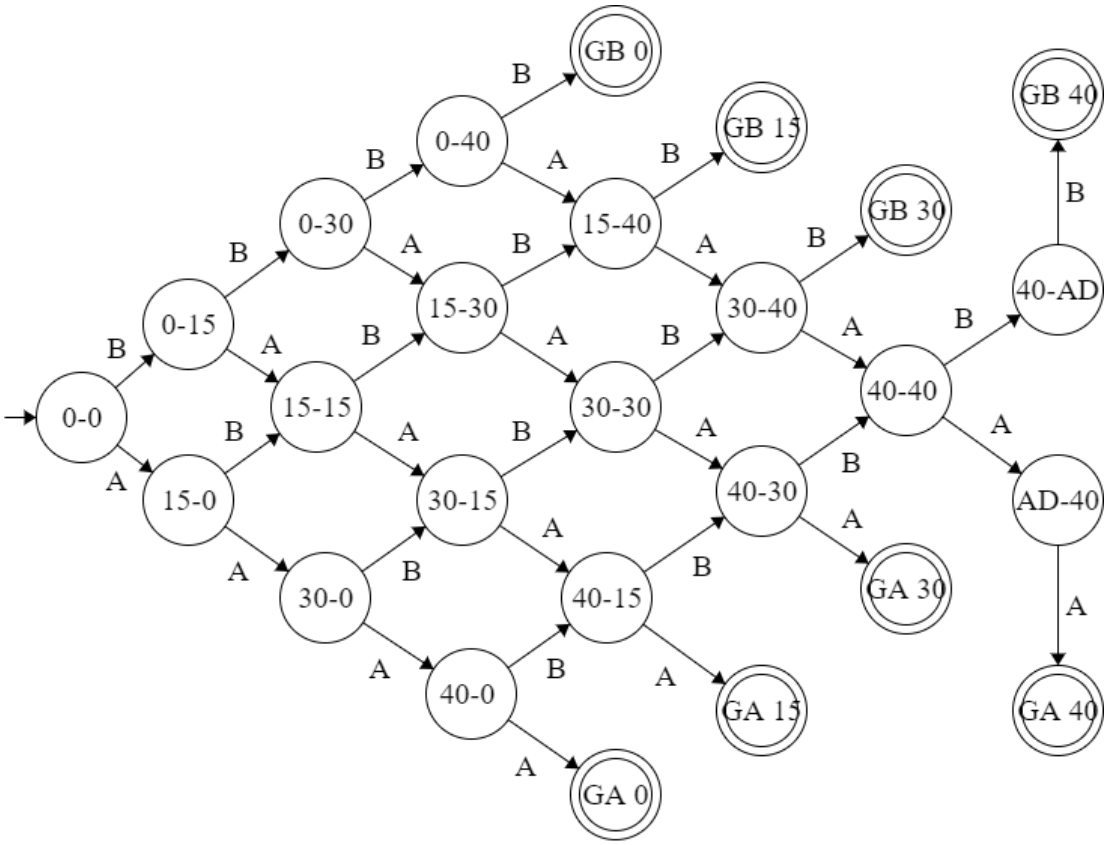
```
Ingresar cadenas de a y b: abababbabab
rejected

Process finished with exit code 0
```

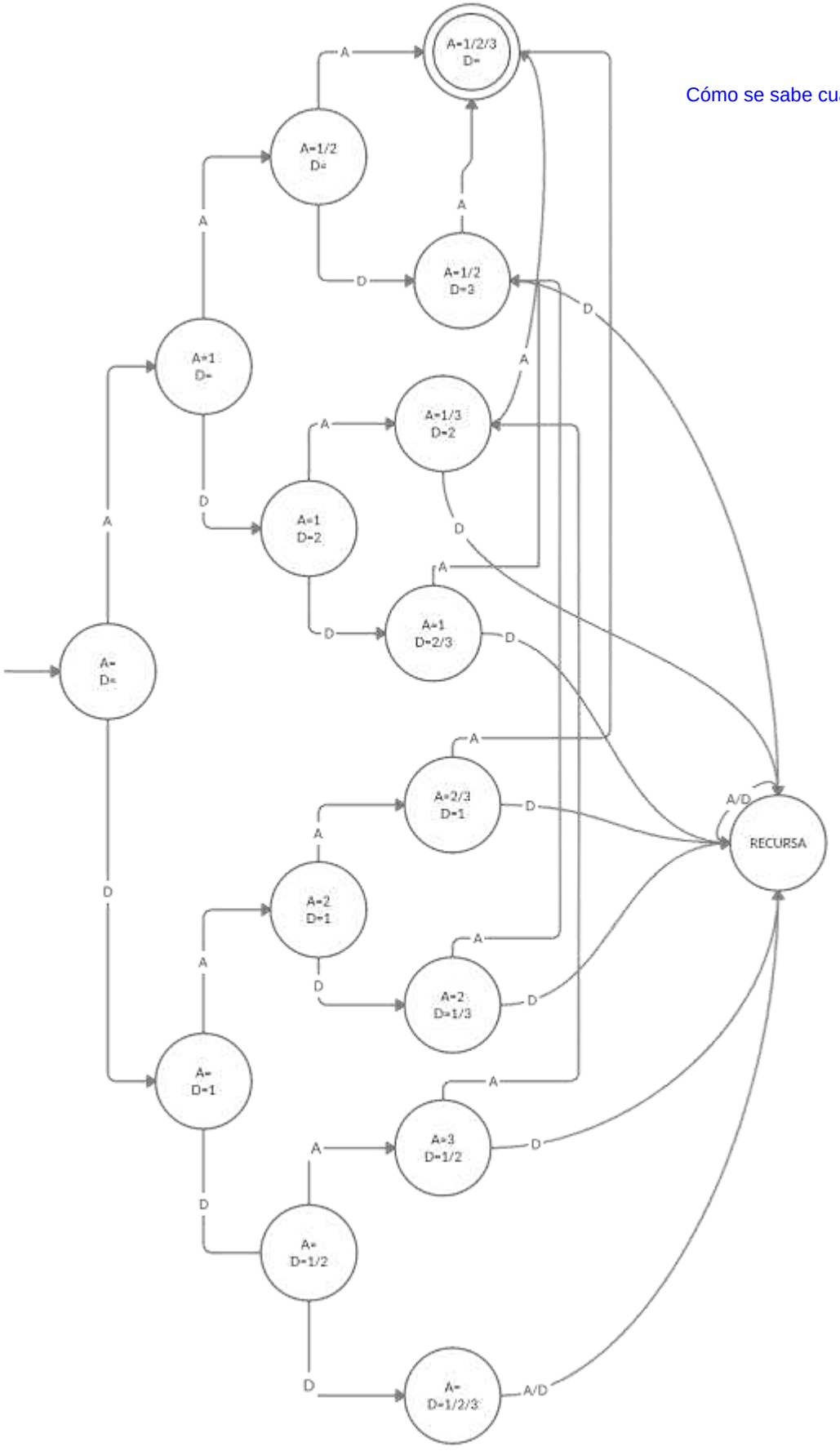
```
Ingresar cadenas de a y b: aaaa
accepted

Process finished with exit code 0
```

1-b)
GA: Ganador Jugador 1, **GB:** Ganador jugador 2
A: Jugador 1, **B**: Jugador 2



1-c)

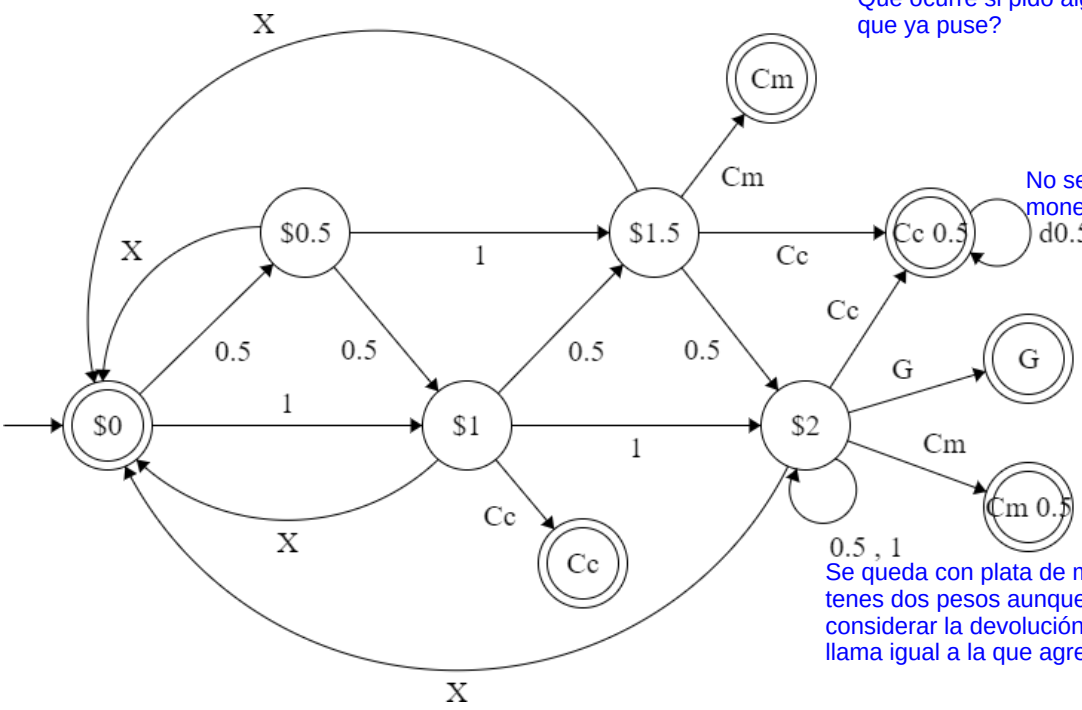


Cómo se sabe cuáles recuperatorios hace?

1.d)

Si cancelo no me devuelve la plata???

Qué ocurre si pido algo que vale menos de lo que ya puse?



No se especifica cuantas monedas devuelve

Se queda con plata de más?? Siempre dice que tenes dos pesos aunque agregues?? Falta considerar la devolución de dinero, la transición se llama igual a la que agrega dinero.

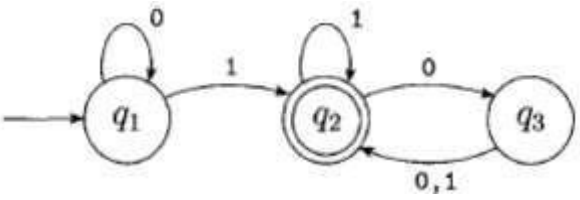
Los estados representan el dinero que tiene la máquina y el producto entregado y su vuelto en caso de ser necesario.
Cc : Cafe chico
Cm : Café medio
G: Gaseosa.
Las transiciones representan el dinero que se ingresa, **X representa la cancelación** y d0.5 es la devolución de $0.5 en caso de que se tenga que devolver más de una vez esa cantidad.

2)

Autómata 1:

a.

Q = { q1, q2, q3}
Σ = {0, 1}
δ =



	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

q0 = {q1} No es un conjunto
F = {q2}

b. L(1) = {*w*|*w* Contiene al menos un 1 y un número par o nula de 0 seguido del último 1}.
L = {1, 01, 011, 1001, 111}.

Autómata 2:

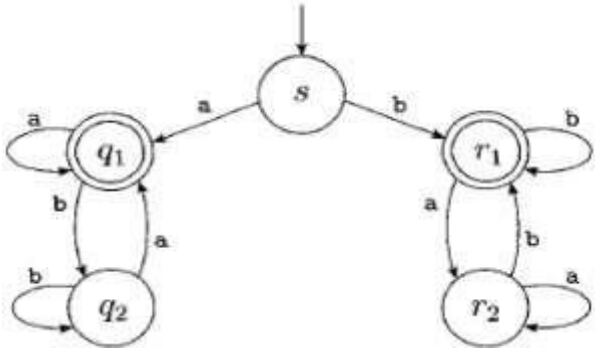
a.

Q = {s, q1, q2, T1, T2}

Σ = {a, b}

δ =

	a	b
s	q1	T1
q1	q1	q2
q2	q1	q2
T1	T2	T1
T2	T2	T1



q0 = {s} No es un conjunto

F = {q1, T1}

b. L(2) = {*w*|*w* Comenzará con el mismo dígito con el que terminará}.

L = {aba, aaba, abba, bab, baabb}.

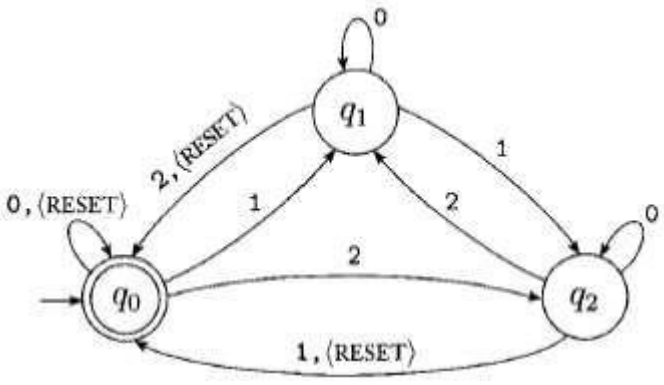
Automata 3:

Q = {q0, q1, q2}

Σ = {0, 1, 2, RESET}

δ =

	0	1	2	RESET
q0	q0	q1	q2	q0
q1	q1	q2	q0	q0
q2	q2	q0	q1	q0



q0 = {q0}

F = {q0}

b. L(3) = {*w*|*w* Contendrá un RESET si la suma de los demás dígitos no es un número divisible por 3}. La suma de los dígitos luego del último RESET es múltiplo de 3

L = {0111, 012, 222, 11RESET, 02RESET}.

Autómata 4:

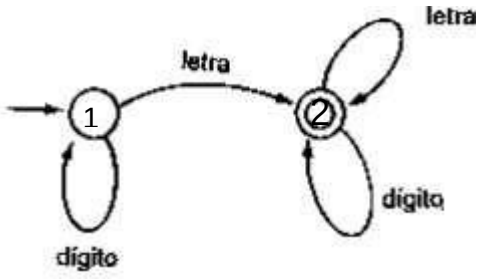
a.

Q = {1, 2 }

Σ = {letra, dígito}

δ =

	letra	dígito
1	2	1
2	2	2



q0 = {1} No es un conjunto

F = {2}

b. L(4) = {*w*|*w* Contendrá al menos una letra}.

L = {letra, digitoletra, letradigito, digitodigitoletra, letraletra}.

Autómata 5:

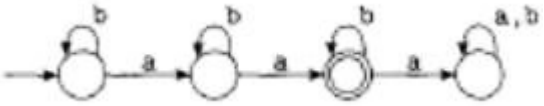
a.

Q = {1, 2, 3, 4}

Σ = {a, b}

δ =

	a	b
1	2	1
2	3	2
3	4	3
4	4	4



q0 = {1} NO es un conjunto

F = {3}

b. L(5) = {*w*|*w* Contendrá si o sí dos a}.

L = {aa, aba, baa, abba, baab, bbaabb}.

Autómata 6:

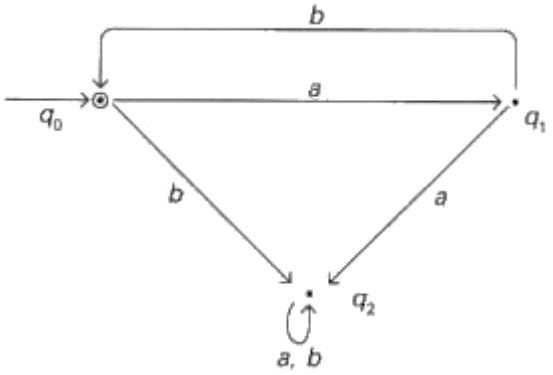
a.

Q = {q0, q1, q2}

Σ = {a, b}

δ =

	a	b
q0	q1	q2
q1	q2	q0
q2	q2	q2



q0 = {q0}

F = {q0}

repeticiones de...

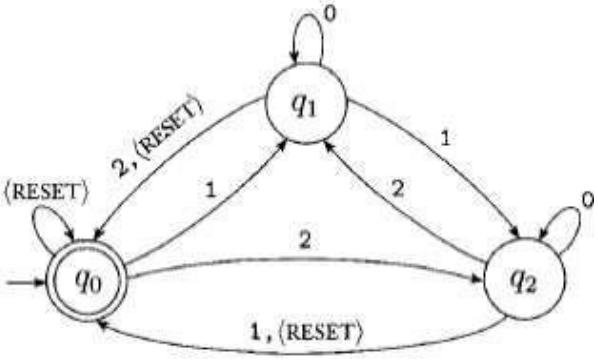
b. L(6) = {*w*|*w* Es la cadena vacía o ab}.

L = {ε, ab}.

3) Toda cadena ingresada terminará en un RESET si la suma de los dígitos anteriores NO es divisible por 3. NO

En caso contrario las cadenas no terminan en RESET si la suma de los dígitos es múltiplo 3.

Despues del último RESET deben ser múltiplo de 3, si no hay RESET también.



Ej:

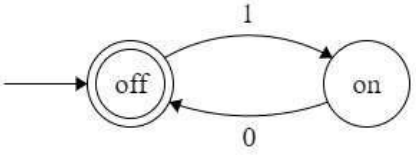
- 0111 → 0+1+1+1 = 3
- 012 → 0+1+2 = 3
- 1122 → 1+1+2+2 = 6

- 11**RESET** → 1+1= 2
- 02**RESET** → 0+2 = 2
- 22**RESET** → 2+2 = 4

4) Switch on/off

1 = Prender

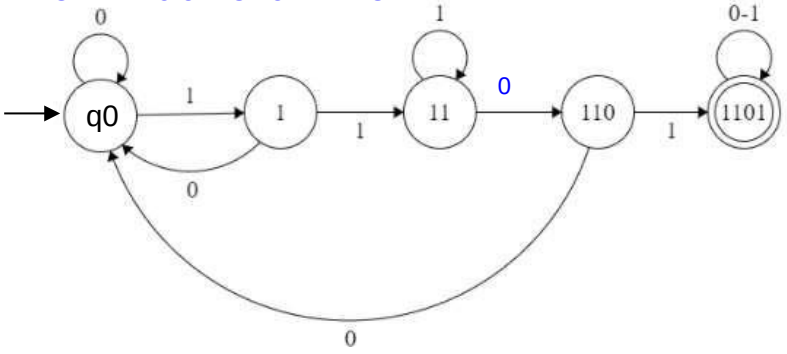
0 = Apagar



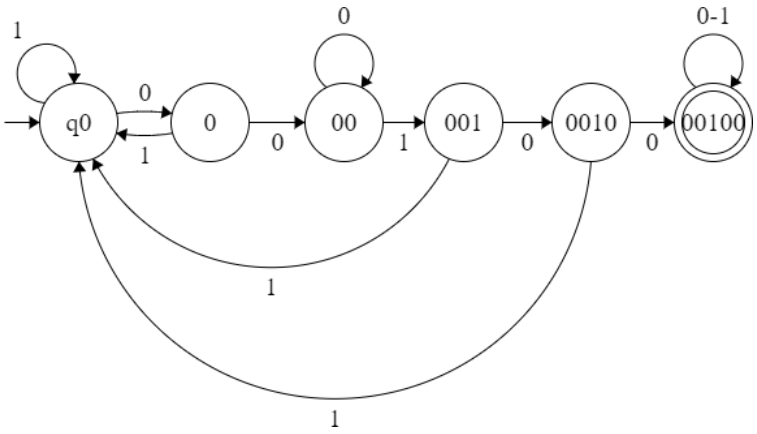
El círculo representa un “0”, y la línea representa un “1”. En efecto, es una mezcla de los números binarios 0 y 1, que representan apagado y **encendido** respectivamente

5) FALTAN LAS DEFINICIONES FORMALES ----------

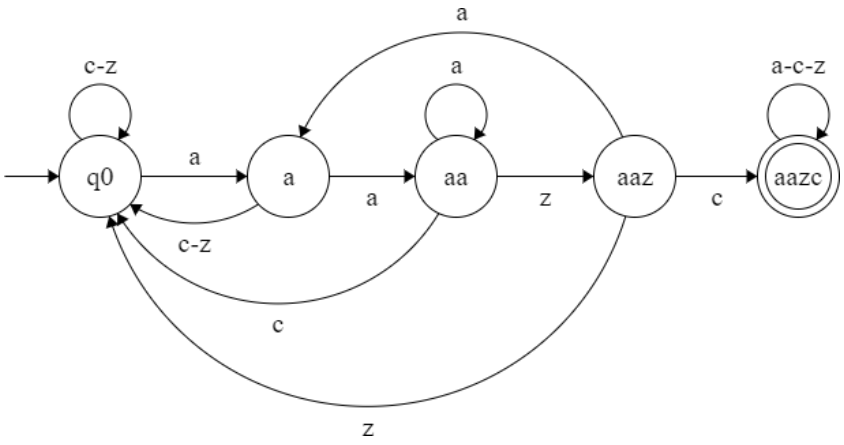
a.



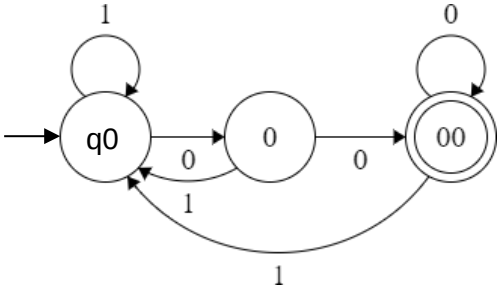
b.



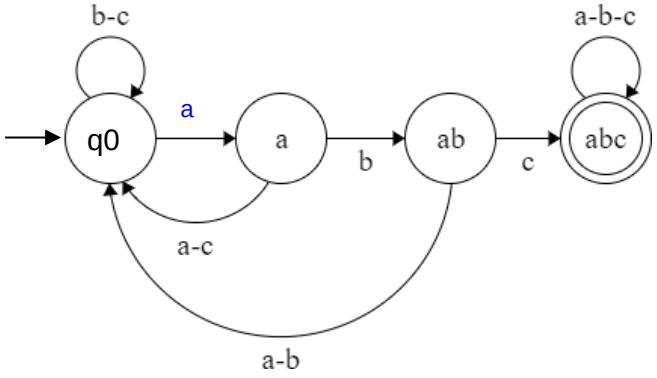
c.



d

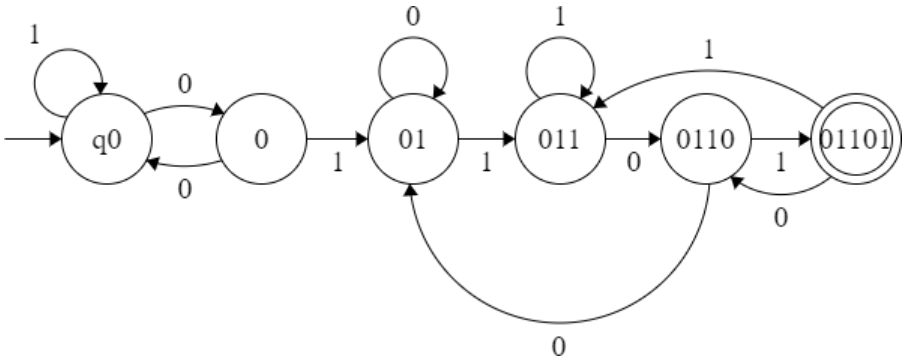


e.



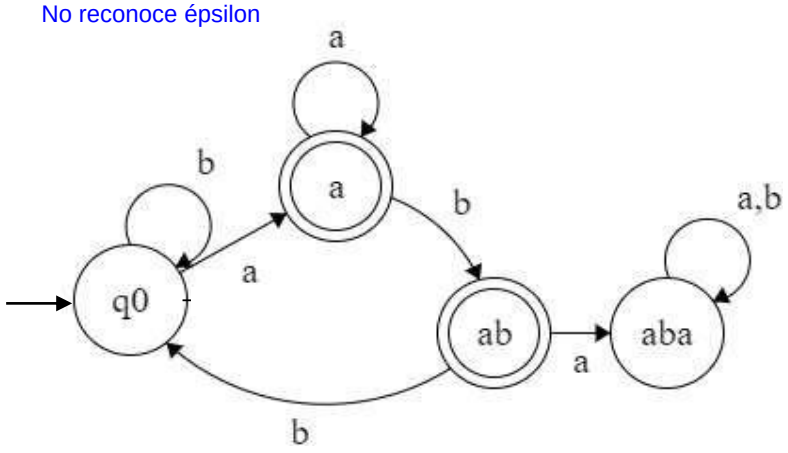
f.

No reconoce la cadena mínima 0101

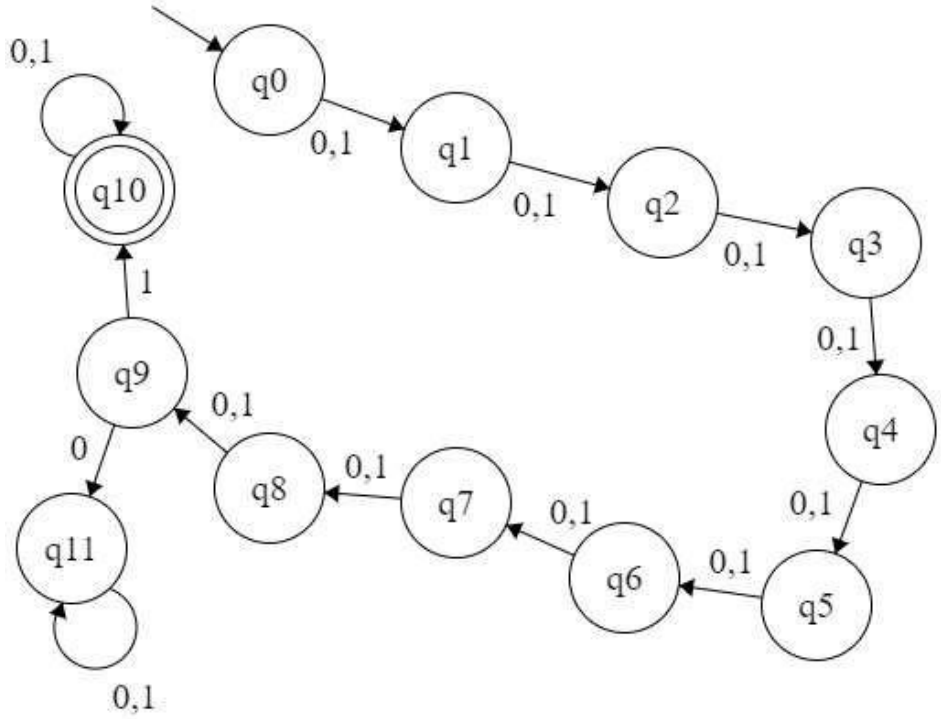


g

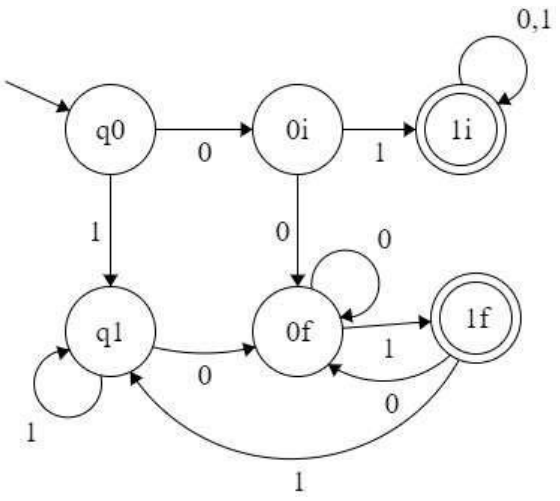
No reconoce épsilon



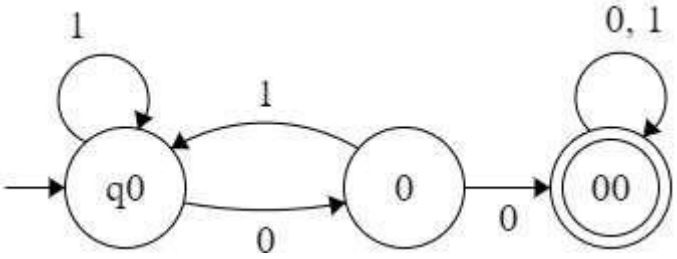
h.



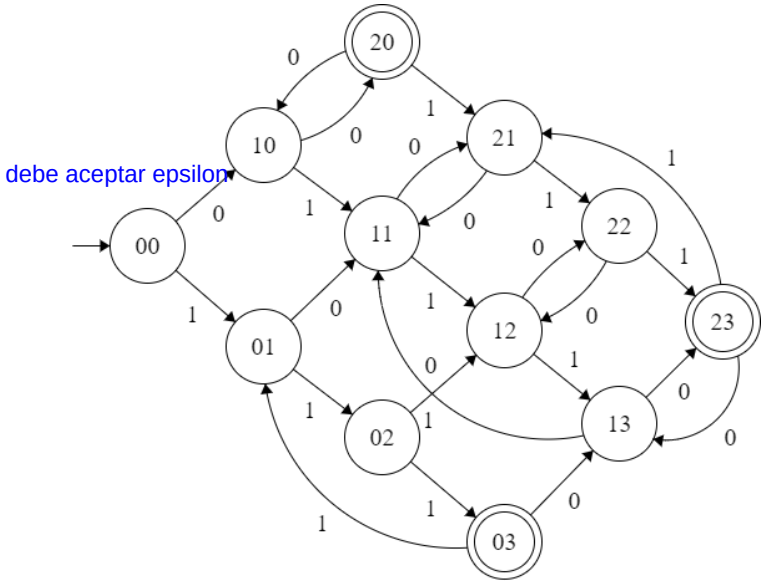
i.



j. Los ceros no necesariamente son consecutivos. Debería reconocer cadenas tales como 010, y no reconocer cadenas como 0010, etc.



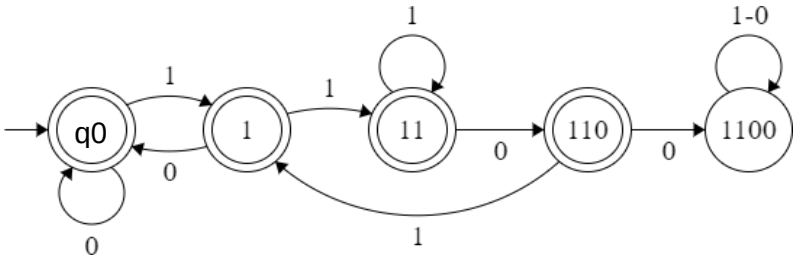
k.



debe aceptar epsilon

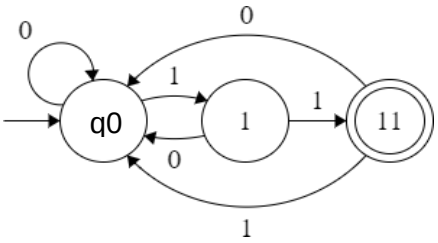
l.

reconoce cadenas inválidas como 110100, etc



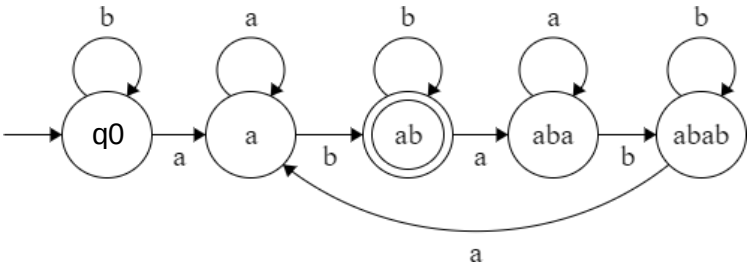
m.

reconoce cadenas que pueden contener más de un par de 1s consecutivos



n

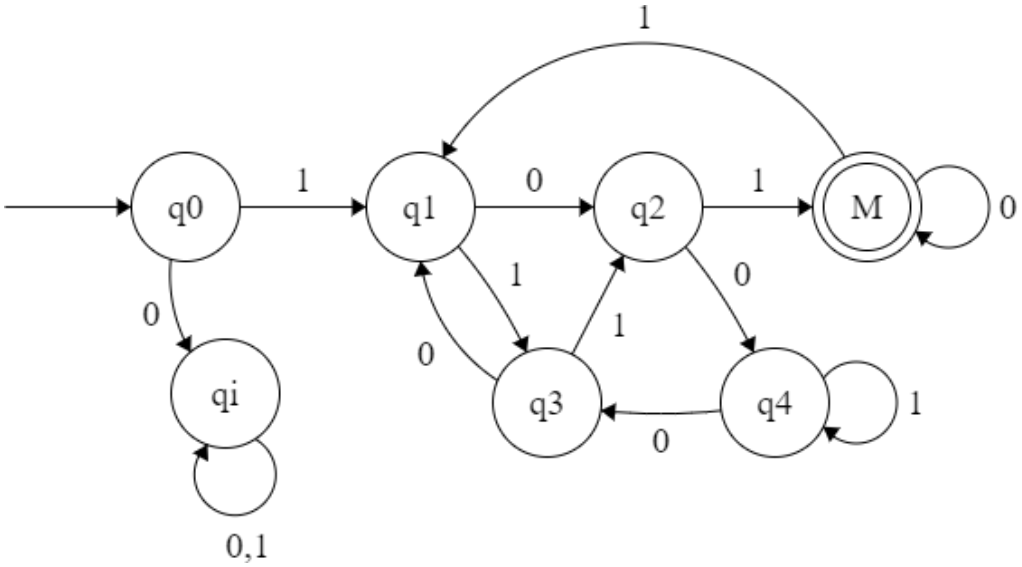
aba también debería ser de aceptación



Parte dos:

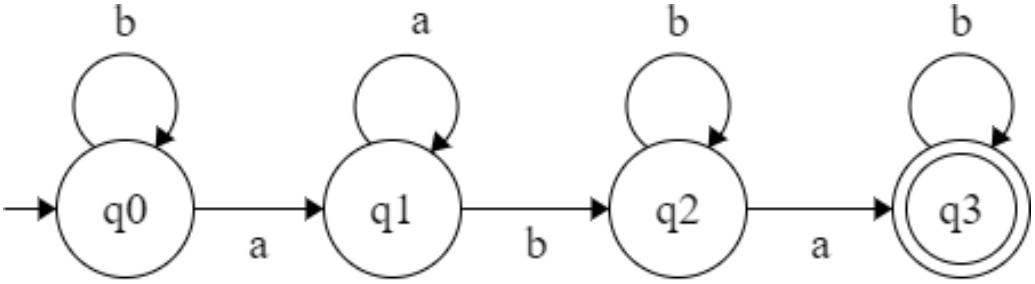
Las demostraciones se encuentran en el archivo Python

6)



Las cadenas que pasen por el estado qi no se tendrán en cuenta ya que no comienzan con 1.
El estado **M** será el final e indicará que un número es múltiplo de 5.

7)

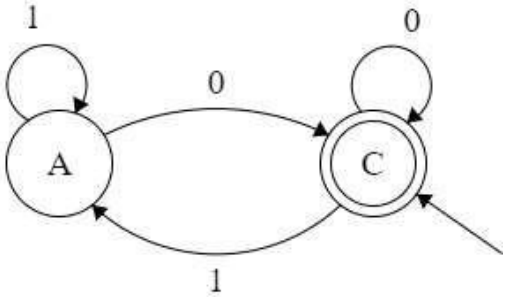


8)

Sistema de puerta automática:

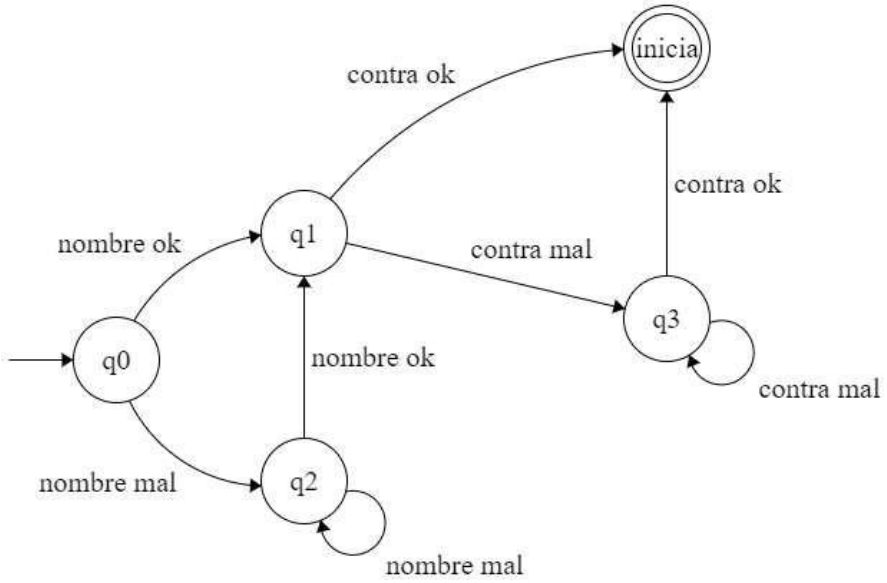
Q = {A,C}
Σ = {0,1}
δ =

	A	C
0	C	C
1	A	A



q0 = {0}
F = {A}

Inicio de sesión de usuario



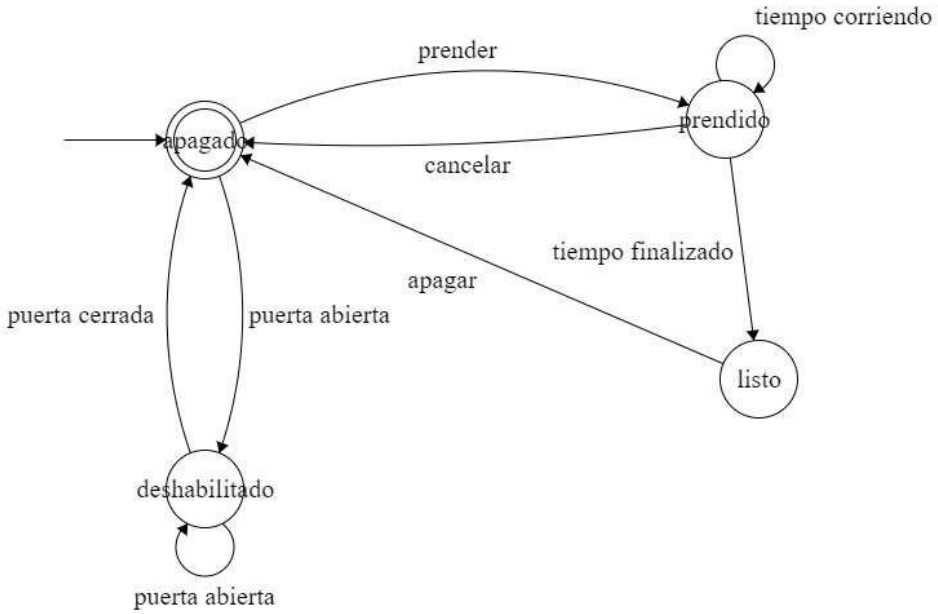
Q = {q0, q1, q2, q3, inicia}
Σ = {nombre ok, nombre mal, contra ok, contra mal}
δ =

ES AFN

	nombre ok	nombre mal	contra ok	contra mal
q0	q1	q2	-	-
q1	-	-	incia	q3
q2	q1	q2	-	-
q3	-	-	incia	q3
incia	-	-	-	-

q0 = {q0}
F = {inicia}

Microondas:



Q = {apagado, deshabilitado, prendido, listo}
Σ = {puerta abierta, puerta cerrada, apagar, prender, cancelar, tiempo corriendo, tiempo finalizado}
δ = CUIDADO, es AFN

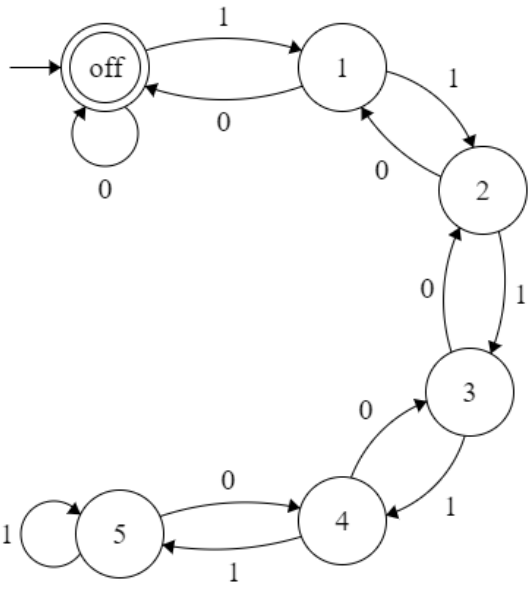
	puerta abierta	puerta cerrada	prender	apagar	cancelar	tiempo corriendo	tiempo finalizado
deshabilitado	deshabilitado	apagado	- ?	- ES AFN....	-	-	-
apagado	deshabilitado	-	prendido	-	-	-	-
prendido	-	-	-	-	apagado	prendido	listo
listo	-	-	-	apagado	-	-	-

q0= {apagado}
F = {apagado}

Perrila 5 velocidades de una licuadora:

Q = {off, 1, 2, 3, 4, 5}
Σ = {0,1}
δ =

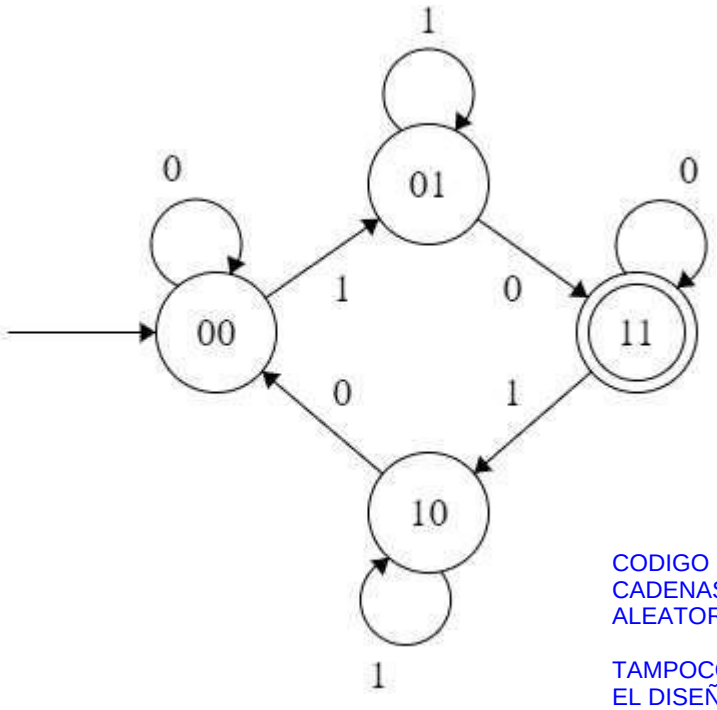
	0	1
off	off	1
1	off	2
2	1	3
3	2	4
4	3	5
5	4	5



q0 = {off}
F = {0ff}

1 representa aumentar la velocidad.
0 representa disminuir velocidad.

9) Parte A:



CODIGO FUENTE: CORRECTO PERO LAS CADENAS NO SON GENERADAS ALEATORIAMENTE POR EL PROGRAMA

TAMPOCO SE VALIDA COMPUTACIONALMENTE EL DISEÑO DEL AUTÓMATA

CORREGIR

Parte B:

a)

File - C:\Users\JUAN CRUZ\PycharmProjects\pythonProject1\sintaxisp3.py

```
1 from automata.fa.dfa import DFA
2 # DFA which matches all binary strings ending in an
  odd number of '1's
3 dfa = DFA(
4     states={'00','01','11','10'},
5     input_symbols={'0', '1'},
6     transitions={
7         '00': {'0': '00', '1': '01'},
8         '01': {'0': '11', '1': '11'},
9         '11': {'0': '11', '1': '10'},
10        '10': {'0': '00', '1': '10'},
11    },
12    initial_state='00',
13    final_states={'11'},
14
15 )
16
17 cadena = input("Ingresar cadenas de 0 o 1 : ")
18
19 if dfa.accepts_input(cadena):
20     print('aceptado')
21 else:
22     print('rechazado')
23
```

b)

Cadenas Aceptadas

```
Ingresar cadenas de 0 o 1 : 1000
aceptado

Process finished with exit code 0
```

Cadenas Rechazadas

```
Ingresar cadenas de 0 o 1 : 1011010010
rechazado

Process finished with exit code 0
```

```
Ingresar cadenas de 0 o 1 : 0111010
aceptado

Process finished with exit code 0
```

```
Ingresar cadenas de 0 o 1 : 1101010010
rechazado

Process finished with exit code 0
```

c)

Verificación si el autómata en el punto “a” es el mínimo:

```
minimal_dfa = dfa.minify()

if(minimal_dfa == dfa):
    print("Es el automata minimo")
else:
    print("No es el automata minimo")
```