

”

**E-fólio A** | Folha de resolução para E-fólio

**UNIDADE CURRICULAR:** Linguagens e Computação

**CÓDIGO:** 21078

**DOCENTES:** Jorge Morais (professor) e Rúdi Gualter (tutor)

**A preencher pelo estudante**

**NOME:** Yrma Marina Vianez Martins

**N.º DE ESTUDANTE:** 2300212

**CURSO:** Engenharia Informática

## TRABALHO / RESOLUÇÃO:

1. Como preciso de números binários o alfabeto será:  $\Sigma = \{0,1\}$

Condição 1: Filho1 só gosta de números binários de tamanho múltiplo de 3, uso o + em vez do \* porque a string vazia  $\epsilon$  não é considerada um número binário.

ER =  $((0|1)(0|1)(0|1))^+$  portanto a sequência é uma sequência de 3 caracteres do alfabeto  $\Sigma = \{0,1\}$ ; \_ \_ \_ em que em cada posição irá ter ou 0 ou 1, e esta sequência repete-se 1 ou mais vezes.

Condição 2: Filho2 só gosta de números binários de tamanho ímpar, mas de valor par (acaba em 0)

ER =  $0|(((0|1)(0|1))^+0)$  Portanto ou é 0 ou é uma sequência do tipo \_ \_ 0, em que os espaços ' \_ ' podem ser 0 ou 1.

(que percebi mais tarde que pode ser apenas  $((0|1)(0|1))^*0$ )

Então o pai pode oferecer aos filhos os números binários que satisfaçam ambas as condições das duas expressões regulares. Necessitam, portanto, de ser números binários de tamanho ímpar múltiplo de 3, mas de valor par.

ER =  $((0|1)(0|1)0) | (((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+)((0|1)(0|1)0)$

A primeira parte até ao '| '  $((0|1)(0|1)0)$  ' representa todas as sequências pares possíveis apenas com 3 dígitos.

A segunda parte representa todas as sequências pares maiores que 3 dígitos múltiplas de 3, de tamanho ímpar, sem incluir a palavra vazia  $\epsilon$ :

' $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+((0|1)(0|1)0)$ '

Portanto repete sempre pelo menos uma vez a sequência de 6 combinações com 0 ou 1, e concatena com a sequência da combinação de 2 (0 ou 1) concatenado com um ultimo 0 para que seja par. Ver ficheiro 1.json

**(nota)** percebi mais tarde que a expressão podia ficar mais simples, mas já tinha feito a maior parte do trabalho, portanto prossegui com a expressão sem ser simplificada. A expressão mais simplificada é:

$((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^*(0|1)(0|1)0$

**2.** Para transformar a expressão regular obtida em um autômato determinístico mínimo, vou fazer em 3 etapas:

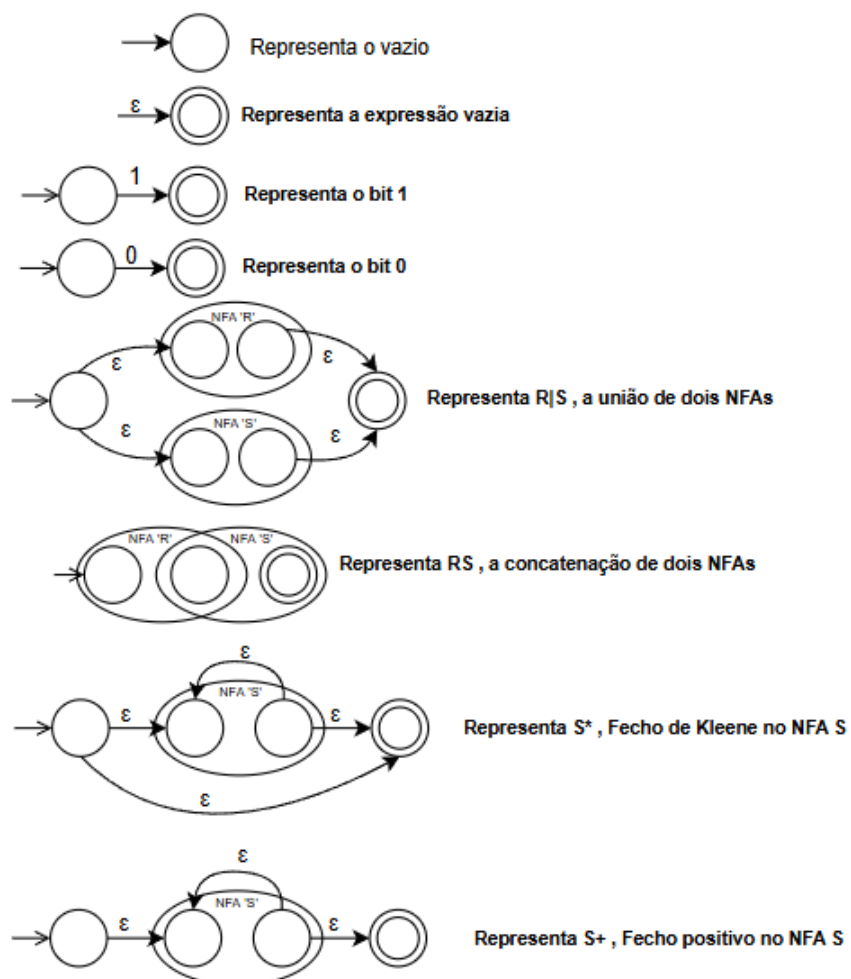
- I. usar o algoritmo de Thompson para obter um NFA- $\epsilon$ 
  - a. Representar o NFA- $\epsilon$  da ER :  $((0|1)(0|1)0)$
  - b. Representar o NFA- $\epsilon$  da ER  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))$
  - c. Representar o NFA- $\epsilon$  da ER:  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+$
  - d. Representar o NFA- $\epsilon$  da ER:  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+ ((0|1)(0|1)0)$
  - e. Representar o NFA- $\epsilon$  da ER final:  $((0|1)(0|1)0) | (((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+ ((0|1)(0|1)0))$

II. Converter o NFA- $\epsilon$  para um DFA, aplicando o algoritmo que utiliza fechos  $\epsilon$  e transições entre estados

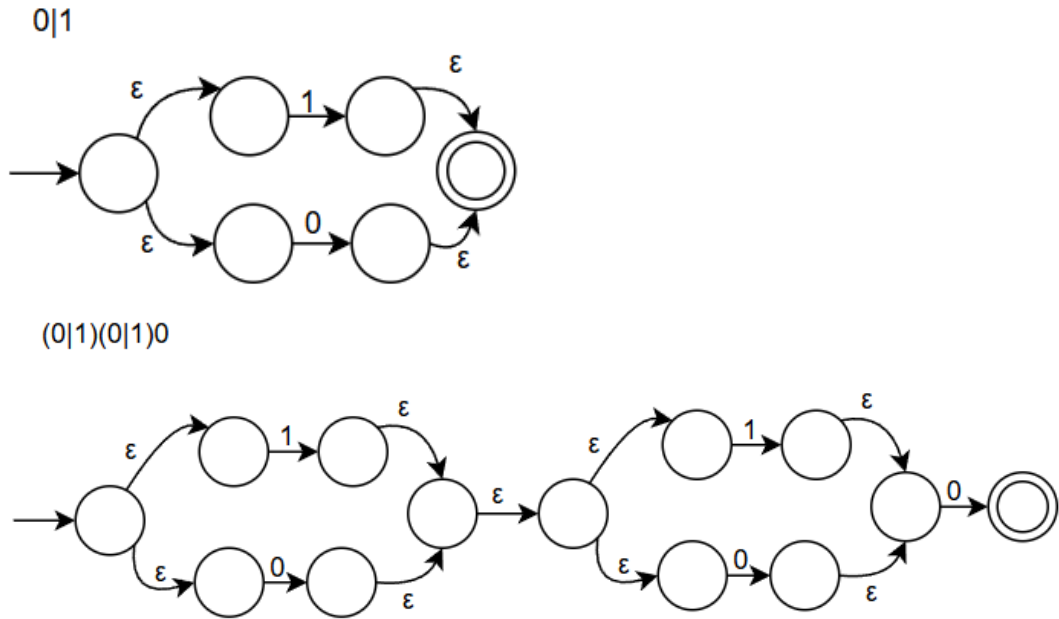
III. minimizar o DFA

I. Primeira etapa: ER  $\rightarrow$  NFA- $\epsilon$

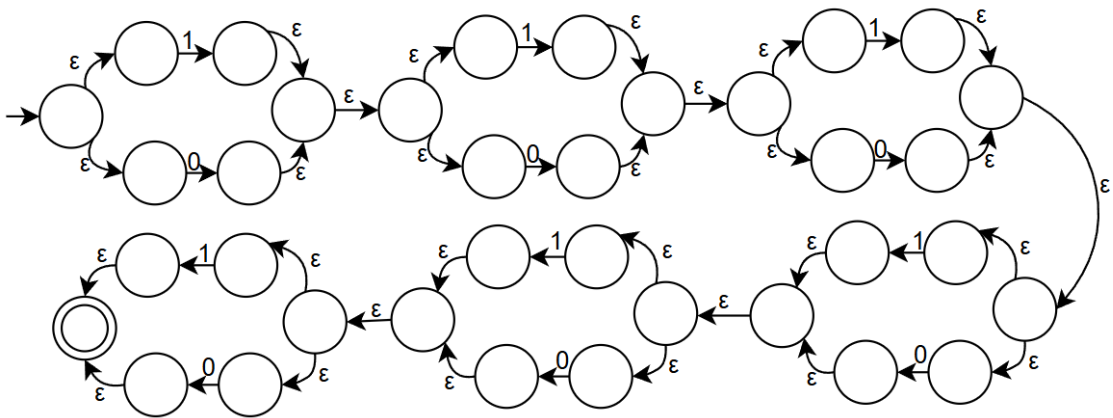
Vou usar as seguintes regras que caracterizam o algoritmo de Thomson:



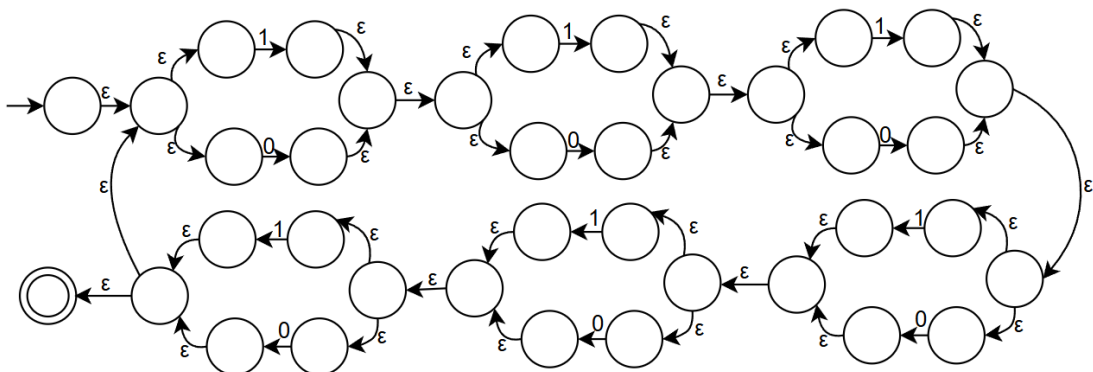
I.a)  $((0|1)(0|1)0)$



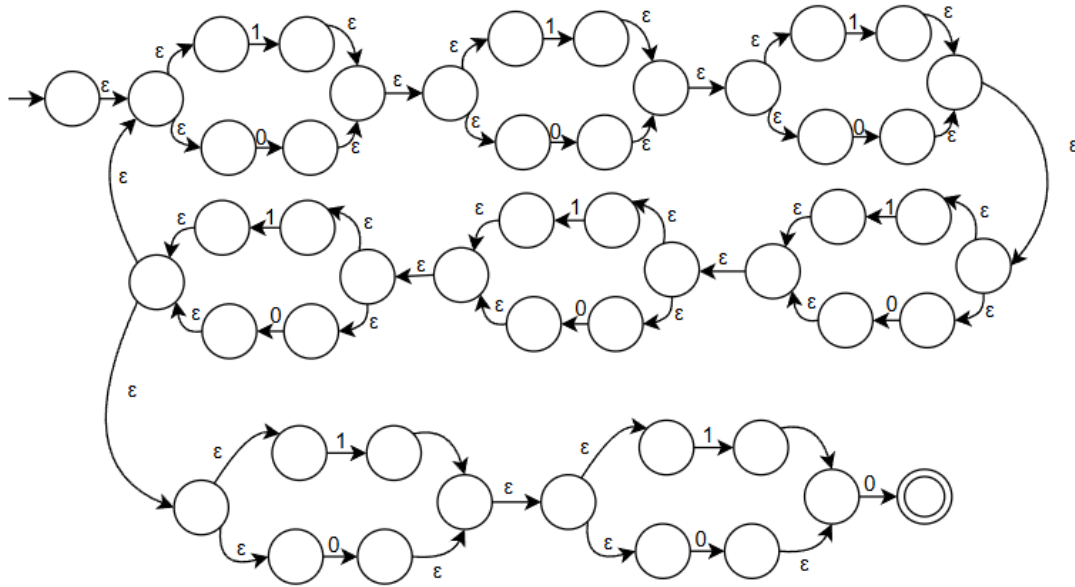
I.b) :  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))$



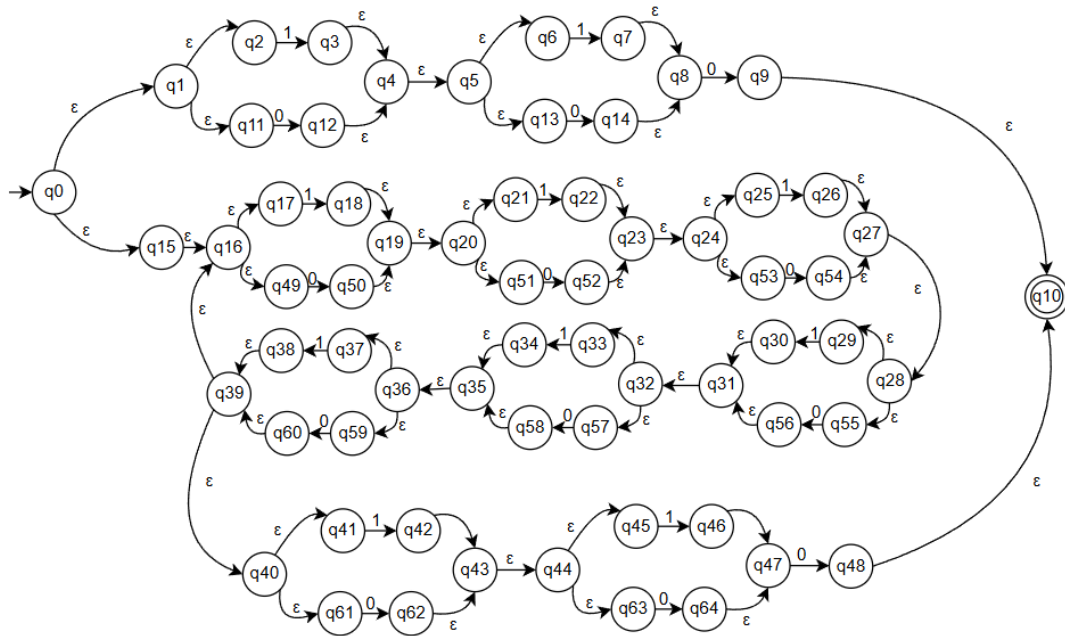
I.c)  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+$



I.d.)  $((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+ ((0|1)(0|1)0)$

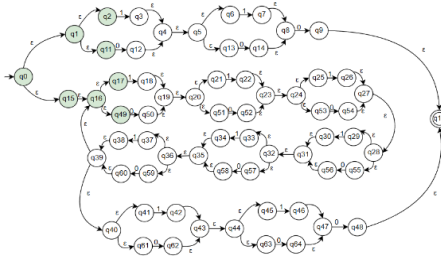


I.e.)  $((0|1)(0|1)0) | (((0|1)(0|1)(0|1)(0|1)(0|1)(0|1))^+ ((0|1)(0|1)0))$

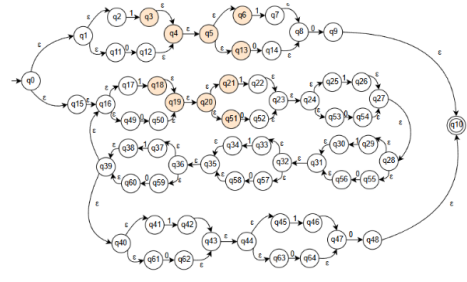
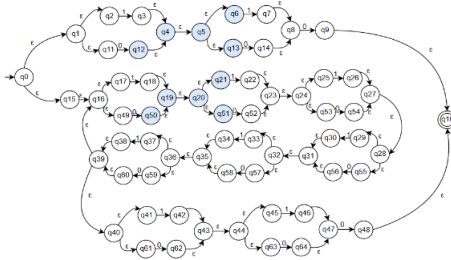


## II. converter NFA- $\epsilon$ para DFA

fecho  $\epsilon (q_0) = \{q_0, q_1, q_2, q_{11}, q_{15}, q_{16}, q_{17}, q_{49}\}$  (estado inicial) ( $\epsilon$ ) = A

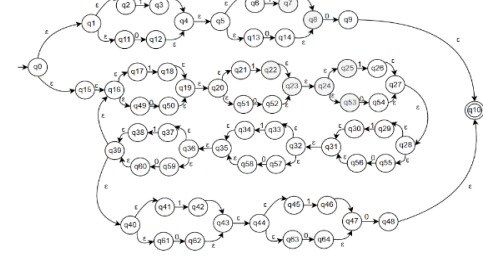
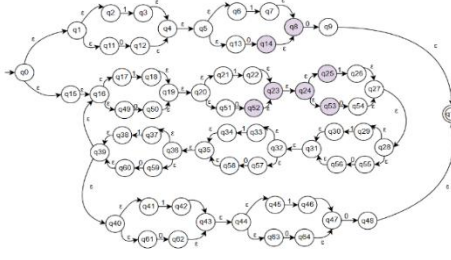


$$\delta(A, 0) = \{q_{12}, q_4, q_5, q_6, q_{13}, q_{50}, q_{19}, q_{20}, q_{21}, q_{51}\} (0) = B \quad \delta(A, 1) = \{q_3, q_4, q_5, q_6, q_{13}, q_{18}, q_{19}, q_{20}, q_{21}, q_{51}\} (1) = C$$



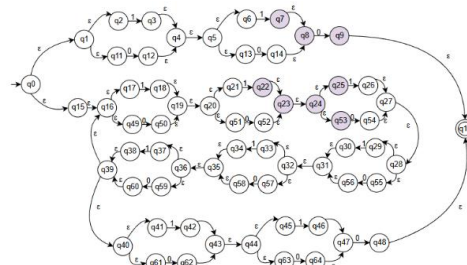
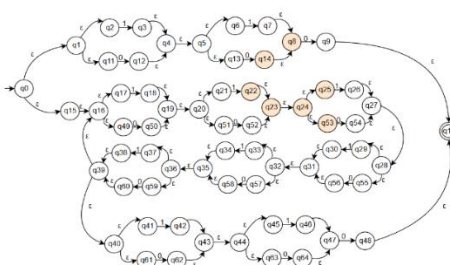
$$\delta(B, 0) = \{q_{14}, q_8, q_{52}, q_{23}, q_{24}, q_{25}, q_{53}\} (00) = D$$

$$\delta(B, 1) = \{q_7, q_8, q_{22}, q_{23}, q_{24}, q_{25}, q_{53}\} (01) = E$$



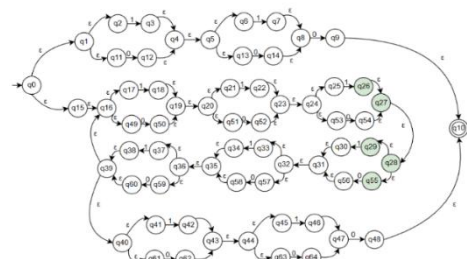
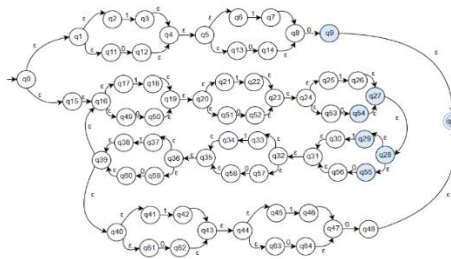
$$\delta(C, 0) = \{q_{14}, q_8, q_{52}, q_{23}, q_{24}, q_{25}, q_{53}\} (10) = D$$

$$\delta(C, 1) = \{q_7, q_8, q_9, q_{22}, q_{23}, q_{24}, q_{25}, q_{53}\} (11) = E$$



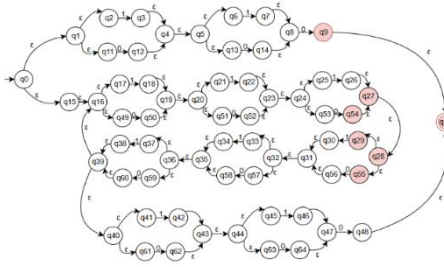
$$\delta(D, 0) = \{q_9, q_{54}, q_{27}, q_{28}, q_{29}, q_{55}\} = F \text{ (estado final)}$$

$$\delta(D, 1) = \{q_{26}, q_{27}, q_{28}, q_{29}, q_{55}\} = G$$

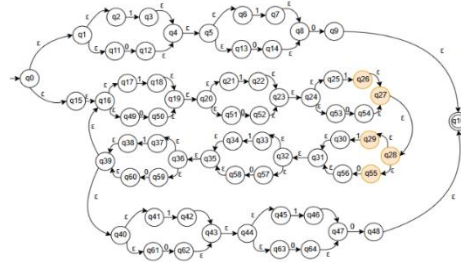




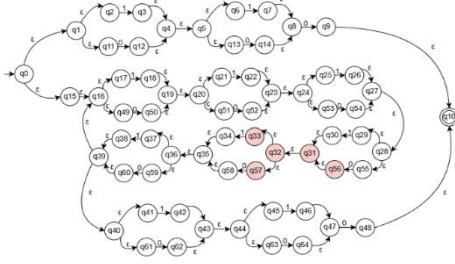
$$\delta(E,0) = \{ q_9, q_{10}, q_{54}, q_{27}, q_{28}, q_{29}, q_{55} \} = F \text{ (estado final)}$$



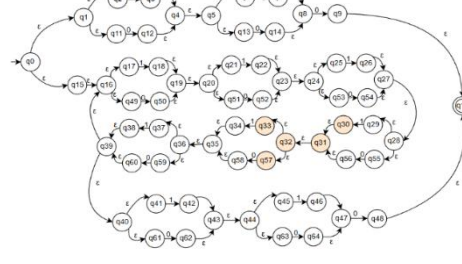
$$\delta(E,1) = \{ q_{26}, q_{27}, q_{28}, q_{29}, q_{55} \} = G$$



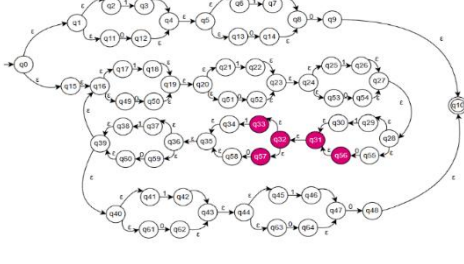
$$\delta(F,0) = \{ q_{56}, q_{31}, q_{32}, q_{33}, q_{57} \} = H$$



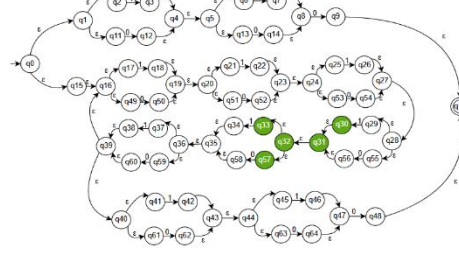
$$\delta(F,1) = \{ q_{30}, q_{31}, q_{32}, q_{33}, q_{57} \} = I$$



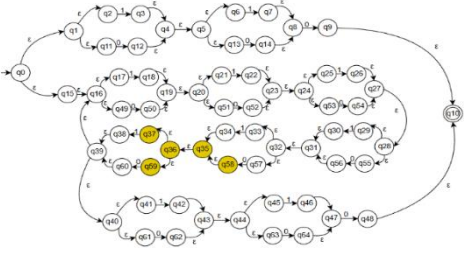
$$\delta(G,0) = \{ q_{56}, q_{31}, q_{32}, q_{33}, q_{57} \} = H$$



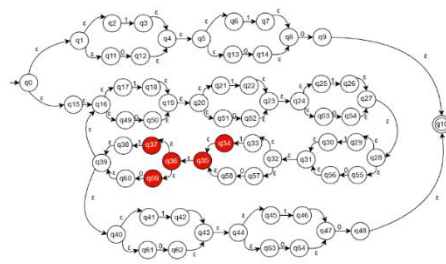
$$\delta(G,1) = \{ q_{30}, q_{31}, q_{32}, q_{33}, q_{57} \} = I$$



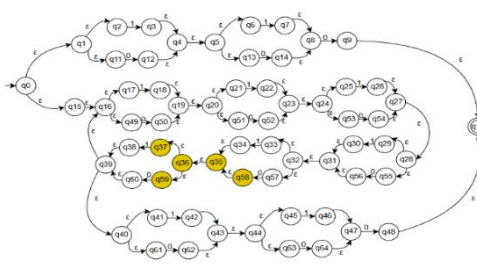
$$\delta(H,0) = \{ q_{58}, q_{35}, q_{36}, q_{37}, q_{59} \} = J$$



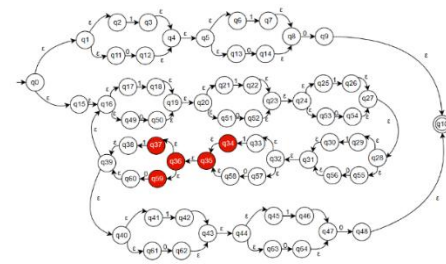
$$\delta(H,1) = \{ q_{34}, q_{35}, q_{36}, q_{37}, q_{59} \} = K$$



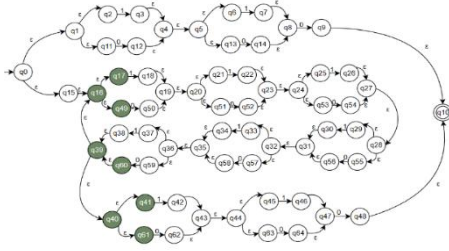
$$\delta(I,0) = \{ q_{58}, q_{35}, q_{36}, q_{37}, q_{59} \} = J$$



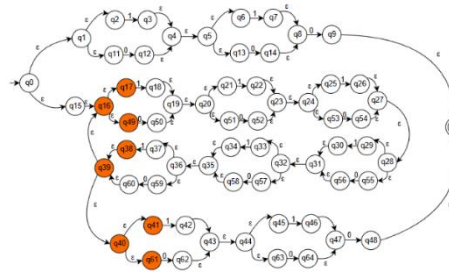
$$\delta(I,1) = \{ q_{34}, q_{35}, q_{36}, q_{37}, q_{59} \} = K$$



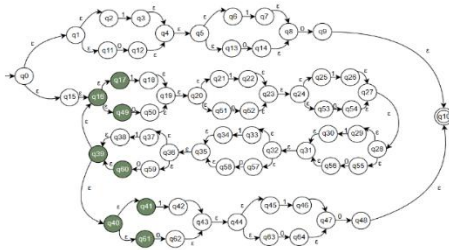
$$\delta(J,0) = \{ q_{60}, q_{39}, q_{40}, q_{41}, q_{61}, q_{16}, q_{17}, q_{49} \} = L$$



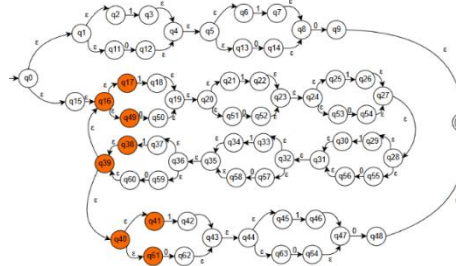
$$\delta(J,1) = \{ q_{38}, q_{39}, q_{40}, q_{41}, q_{61}, q_{16}, q_{17}, q_{49} \} = M$$



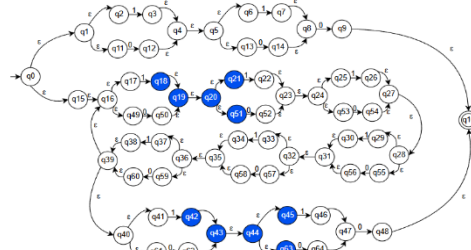
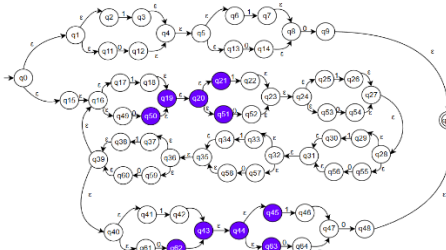
$$\delta(K,0) = \{ q_{60}, q_{39}, q_{40}, q_{41}, q_{61}, q_{16}, q_{17}, q_{49} \} = L$$



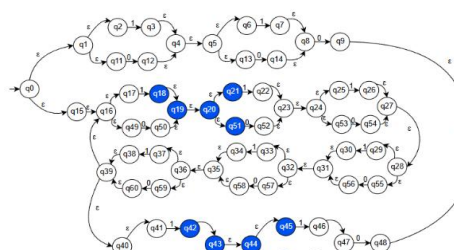
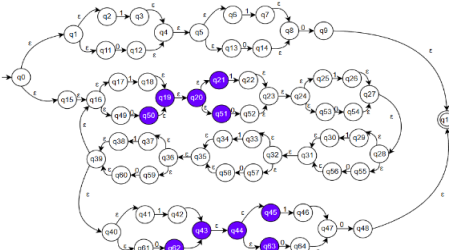
$$\delta(K,1) = \{ q_{38}, q_{39}, q_{40}, q_{41}, q_{61}, q_{16}, q_{17}, q_{49} \} = M$$



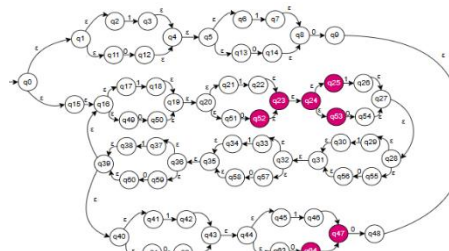
$$\delta(L,0) = \{ q_{50}, q_{19}, q_{20}, q_{21}, q_{51}, q_{62}, q_{43}, q_{44}, q_{45}, q_{63} \} = N \quad \delta(L,1) = \{ q_{50}, q_{19}, q_{20}, q_{21}, q_{51}, q_{62}, q_{43}, q_{44}, q_{45}, q_{63} \} = O$$



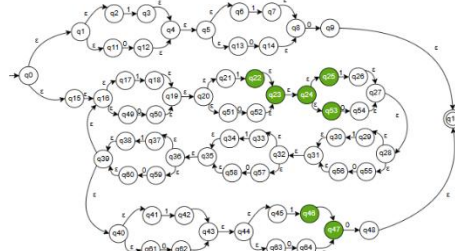
$$\delta(M,0) = \{ q_{50}, q_{19}, q_{20}, q_{21}, q_{51}, q_{62}, q_{43}, q_{44}, q_{45}, q_{63} \} = N \quad \delta(M,1) = \{ q_{50}, q_{19}, q_{20}, q_{21}, q_{51}, q_{62}, q_{43}, q_{44}, q_{45}, q_{63} \} = O$$



$$\delta(N,0) = \{ q_{64}, q_{47}, q_{52}, q_{23}, q_{24}, q_{25}, q_{53} \} = P$$

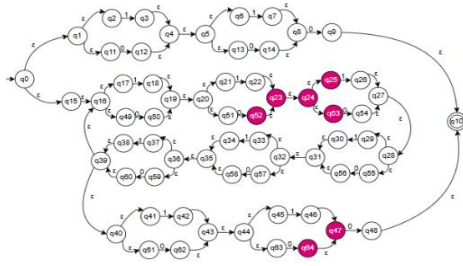


$$\delta(N,1) = \{ q_{46}, q_{47}, q_{22}, q_{23}, q_{24}, q_{25}, q_{53} \} = Q$$

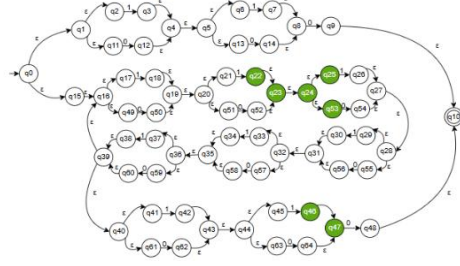




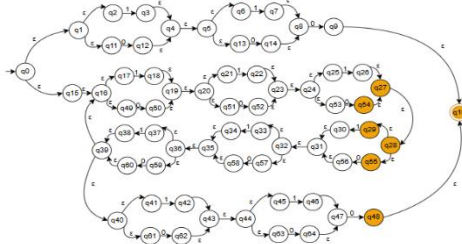
$$\delta(O,0) = \{ q_{64}, q_{47}, q_{52}, q_{23}, q_{24}, q_{25}, q_{53} \} = P$$



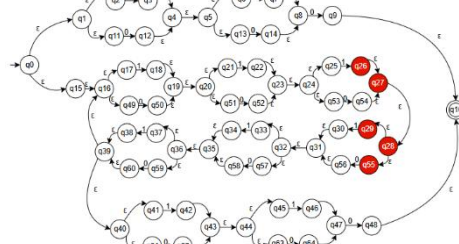
$$\delta(O,1) = \{ q_{46}, q_{47}, q_{22}, q_{23}, q_{24}, q_{25}, q_{53} \} = Q$$



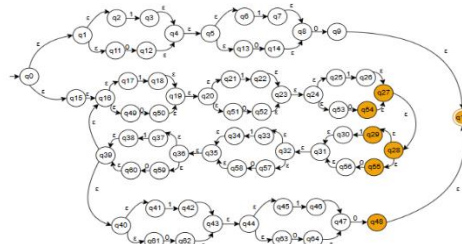
$$\delta(P,0) = \{ q_{48}, q_{10}, q_{54}, q_{27}, q_{28}, q_{29}, q_{55} \} = R(\text{estado final})$$



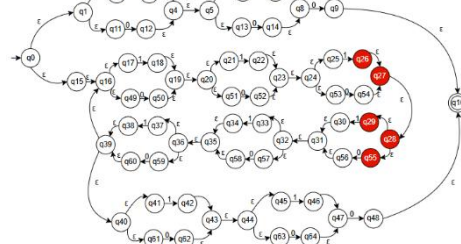
$$\delta(P,1) = \{ q_{26}, q_{27}, q_{28}, q_{29}, q_{55} \} = G$$



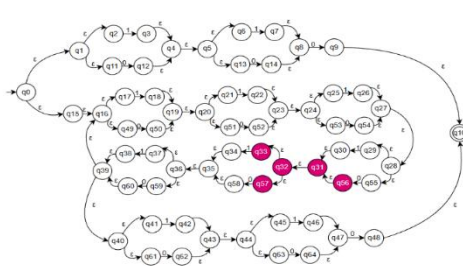
$$\delta(Q,0) = \{ q_{48}, q_{10}, q_{54}, q_{27}, q_{28}, q_{29}, q_{55} \} = R(\text{estado final})$$



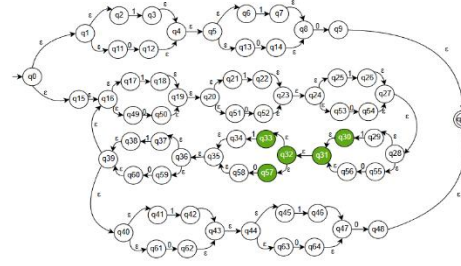
$$\delta(Q,1) = \{ q_{26}, q_{27}, q_{28}, q_{29}, q_{55} \} = G$$



$$\delta(R,0) = \{ q_{56}, q_{31}, q_{32}, q_{33}, q_{57} \} = H$$



$$\delta(R,1) = \{ q_{30}, q_{31}, q_{32}, q_{33}, q_{57} \} = I$$



### III minimizar o DFA

Representei os resultados numa tabela de transições para facilitar a comparação e verificar estados obsoletos.

Estados	0	1
→A	B	C
B	D	E
C	D	E
D	*F	G
E	*F	G
*F	H	I
G	H	I
H	J	K
I	J	K
J	L	M
K	L	M
L	N	O
M	N	O
N	P	Q
O	P	Q
P	*R	G
Q	*R	G
*R	H	I

Estados	0	1	
→A	B	C	A
B	D	E	B
C	D	E	D
D	*F	G	
E	*F	G	
*F	H	I	*F
G	H	I	G
H	J	K	H
I	J	K	J
J	L	M	L
K	L	M	N
L	N	O	P
M	N	O	
N	P	Q	
O	P	Q	
P	*R	G	
Q	*R	G	
*R	H	I	*R

Estados	0	1	
→A	B	B	
B	D	D	
D	*F	G	D
*F	H	H	
G	H	H	
H	J	J	
J	L	L	
L	N	N	
N	P	P	
P	*F	G	

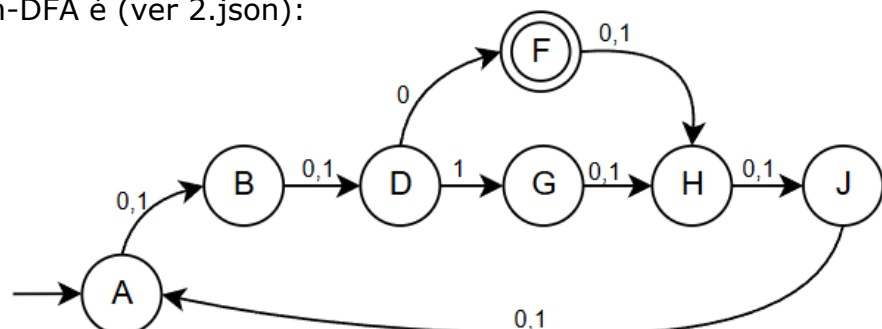
Facilmente verifiquei que tenho vários estados equivalentes ( $B=C$ ;  $D=E$ ;  $F=R$ ;  $H=I$ ;  $J=K$ ;  $L=M$ ;  $N=O$ ), sublinhados na tabela acima. Fiz a tabela com as alterações (tabela acima á direita), e voltei a 2 verificar estados equivalentes ( $D=P$ ), alterei (tabela a baixo á esquerda) e após essa alteração fiquei novamente com um outro estado equivalente( $B=N$ ), e depois novamente  $A = L$ (tabela abaixo no centro). Por fim obtive a tabela de transição minimizada (tabela da esquerda).

Estados	0	1
→A	B	B
B	D	D
D	*F	G
*F	H	H
G	H	H
H	J	J
J	L	L
L	N	N
N	D	D

Estados	0	1	
→A	B	B	
B	D	D	
D	*F	G	
*F	H	H	
G	H	H	
H	J	J	
J	L	L	
L	B	B	A

Estados	0	1
→A	B	B
B	D	D
D	*F	G
*F	H	H
G	H	H
H	J	J
J	A	A

Assim o min-DFA é (ver 2.json):



**3.** Como já escrevi no ponto 1 para satisfazer o filho 1 tem que oferecer números binários de tamanho múltiplo de 3, e, portanto, satisfaçam a seguinte Expressão regular:

$$ER = ((0|1)(0|1)(0|1))^+$$

Para satisfazer o filho 2 tem que oferecer números binários de tamanho ímpar, mas de valor par (acaba em 0), e, portanto, satisfaçam a seguinte Expressão regular:

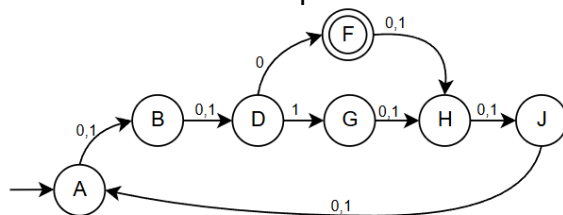
$$ER = 0|(((0|1)(0|1))^+0) \text{ que posso reduzir para } (((0|1)(0|1))^*0)$$

Para satisfazer pelo menos 1 dos filhos basta fazer a união das duas expressões regulares:

$$(((0|1)(0|1)(0|1))^+) \mid (((0|1)(0|1))^*0) \text{ (ver ficheiro 3.json)}$$

4. Como já tenho o DFA de uma parte da expressão feita, pois a questão 1 era a interceção dos gostos dos 2 irmãos, e esse DFA está incluído na união dos gostos dos dois filhos, posso então partir desse DFA para chegar ao da nova expressão.

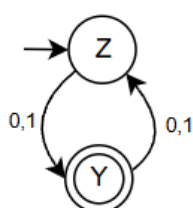
Este aceita os números binários de tamanho ímpar múltiplo de 3 e de valor par. Mas agora além desses números deve também aceitar os números pares de tamanho ímpar e todos de tamanho múltiplo de 3.



Vou alterar por partes:

1- Aceitar números pares de tamanho ímpar:

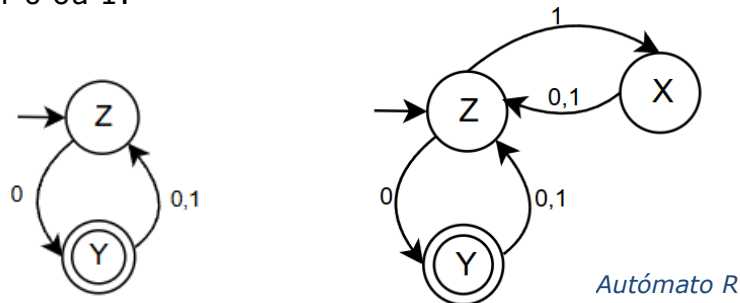
Comecei por imaginar como seria o DFA para aceitar 1 número de tamanho ímpar, e é algo muito simples:



O estado Z trata todos os números de tamanho par

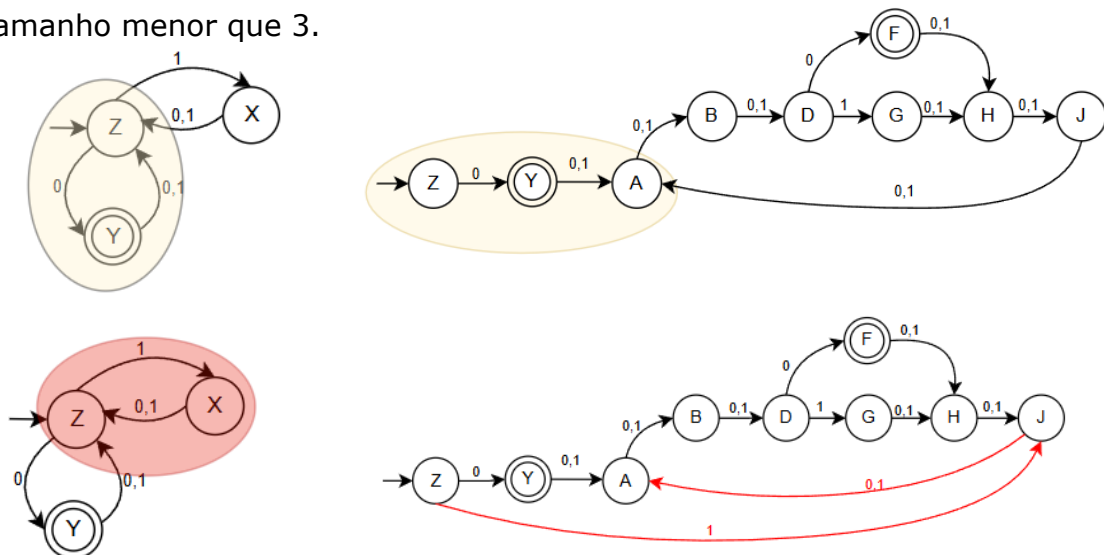
O estado Y (estado de aceitação) trata os de tamanho ímpar.

E depois como quero apenas os números pares de tamanho ímpar, a transição 1 de Z para Y deixa de ser aceite. Necessito, portanto, criar um outro estado para a transição 1 que irá voltar para Z independentemente da transição for 0 ou 1:



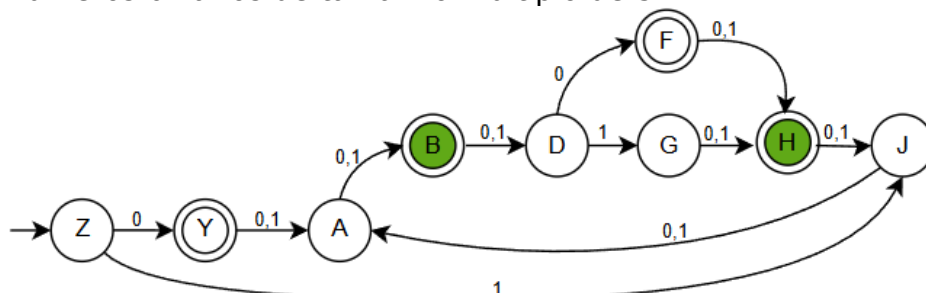
Vou denominar o autômato de cima à direita de autômato R, para ser mais fácil referir-me a ele.

Faz sentido agora, alterar o estado inicial e começar a partir de algo equivalente pois no DFA da questão 2 não aceita números binários de tamanho menor que 3.

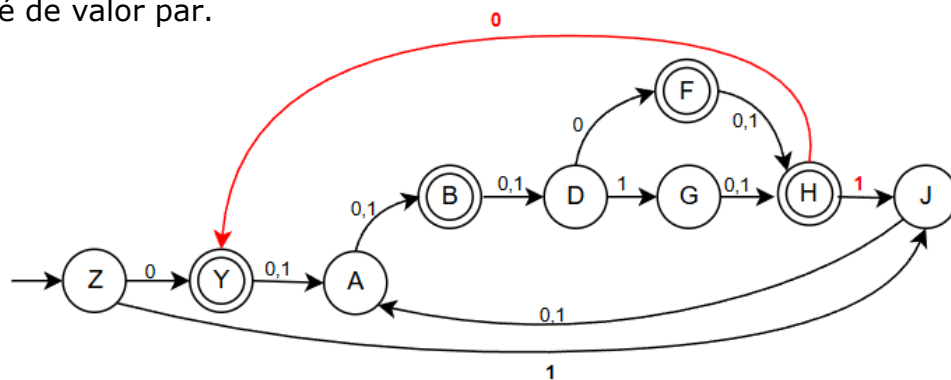


Portanto o estado A do autômato da questão 2 funciona como o estado Z do autômato R e o estado J vai funcionar como o estado X.

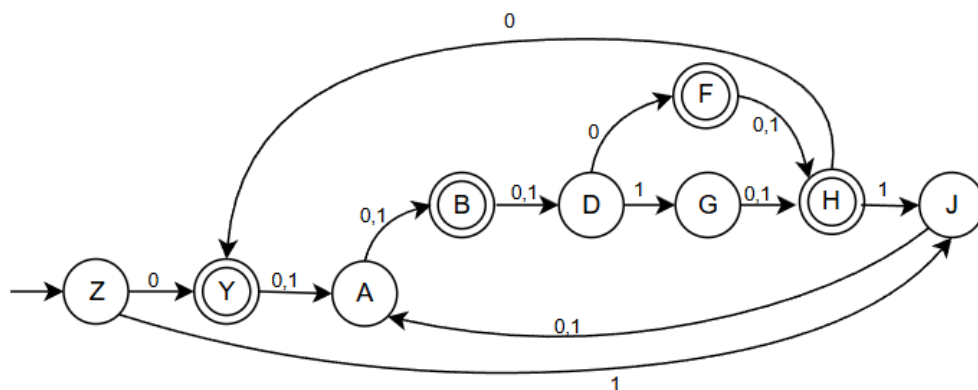
2- Mas agora preciso ajustar os estados finais para poder preservar os números binários de tamanho múltiplo de 3



E ajustar as transições do novo estado final H, pois como nesse estado a transição para 0 é um número binário par de tamanho ímpar, que é aceite, mas a transição para 1 não é aceite porque apesar do tamanho ser ímpar não é de valor par.



Então o DFA min para a expressão regular da questão 3, que representa os números binários que satisfazem pelo menos 1 dos filhos é, (ver ficheiro 4.json):



**5.** Os ficheiros pedidos estão nomeados de 1 a 4 de acordo com as questões do efolio.

- Resultado obtido dos testes de todas as sequências possíveis do alfabeto de tamanho inferior a 10 que obtiveram sucesso no ponto 3 mas insucesso no ponto 1. Pode ser consultado no ficheiro insu1\_scu3.xl.

1

0	Unsuccessful!
001	Unsuccessful!
101	Unsuccessful!
011	Unsuccessful!
111	Unsuccessful!
00000	Unsuccessful!
10000	Unsuccessful!
01000	Unsuccessful!
11000	Unsuccessful!
00100	Unsuccessful!
10100	Unsuccessful!
01100	Unsuccessful!
11100	Unsuccessful!
00010	Unsuccessful!
10010	Unsuccessful!
01010	Unsuccessful!
11010	Unsuccessful!
00110	Unsuccessful!
10110	Unsuccessful!
01110	Unsuccessful!
11110	Unsuccessful!
000000	Unsuccessful!
100000	Unsuccessful!
010000	Unsuccessful!
110000	Unsuccessful!
001000	Unsuccessful!
101000	Unsuccessful!
011000	Unsuccessful!
111000	Unsuccessful!
000100	Unsuccessful!
100100	Unsuccessful!
010100	Unsuccessful!
110100	Unsuccessful!
001100	Unsuccessful!
101100	Unsuccessful!
011100	Unsuccessful!
111100	Unsuccessful!
000010	Unsuccessful!
100010	Unsuccessful!
010010	Unsuccessful!
110010	Unsuccessful!
001010	Unsuccessful!

3

0	Successful!
001	Successful!
101	Successful!
011	Successful!
111	Successful!
00000	Successful!
10000	Successful!
01000	Successful!
11000	Successful!
00100	Successful!
10100	Successful!
01100	Successful!
11100	Successful!
00010	Successful!
10010	Successful!
01010	Successful!
11010	Successful!
00110	Successful!
10110	Successful!
01110	Successful!
11110	Successful!
000000	Successful!
100000	Successful!
010000	Successful!
110000	Successful!
001000	Successful!
101000	Successful!
011000	Successful!
111000	Successful!
000100	Successful!
100100	Successful!
010100	Successful!
110100	Successful!
001100	Successful!
101100	Successful!
011100	Successful!
111100	Successful!
000010	Successful!
100010	Successful!
010010	Successful!
110010	Successful!
001010	Successful!

1

011010	Unsuccessful!
111010	Unsuccessful!
000110	Unsuccessful!
100110	Unsuccessful!
010110	Unsuccessful!
110110	Unsuccessful!
001110	Unsuccessful!
101110	Unsuccessful!
011110	Unsuccessful!
111110	Unsuccessful!
000001	Unsuccessful!
100001	Unsuccessful!
010001	Unsuccessful!
110001	Unsuccessful!
001001	Unsuccessful!
101001	Unsuccessful!
011001	Unsuccessful!
111001	Unsuccessful!
000101	Unsuccessful!
100101	Unsuccessful!
010101	Unsuccessful!
110101	Unsuccessful!
001010	Unsuccessful!
101010	Unsuccessful!
011010	Unsuccessful!
111010	Unsuccessful!
000011	Unsuccessful!
100011	Unsuccessful!
010011	Unsuccessful!
110011	Unsuccessful!
001011	Unsuccessful!
101011	Unsuccessful!
011011	Unsuccessful!
111011	Unsuccessful!
000111	Unsuccessful!
100111	Unsuccessful!
010111	Unsuccessful!
110111	Unsuccessful!
001111	Unsuccessful!
101111	Unsuccessful!
011111	Unsuccessful!
111111	Unsuccessful!
0000000	Unsuccessful!
1000000	Unsuccessful!
0100000	Unsuccessful!

3

011010	Successful!
111010	Successful!
000110	Successful!
100110	Successful!
010110	Successful!
110110	Successful!
001110	Successful!
101110	Successful!
011110	Successful!
111110	Successful!
000001	Successful!
100001	Successful!
010001	Successful!
110001	Successful!
001001	Successful!
101001	Successful!
011001	Successful!
111001	Successful!
000101	Successful!
100101	Successful!
010101	Successful!
110101	Successful!
001010	Successful!
101010	Successful!
011010	Successful!
111010	Successful!
000011	Successful!
100011	Successful!
010011	Successful!
110011	Successful!
001011	Successful!
101011	Successful!
011011	Successful!
111011	Successful!
000111	Successful!
100111	Successful!
010111	Successful!
110111	Successful!
001111	Successful!
101111	Successful!
011111	Successful!
111111	Successful!
0000000	Successful!
1000000	Successful!
0100000	Successful!







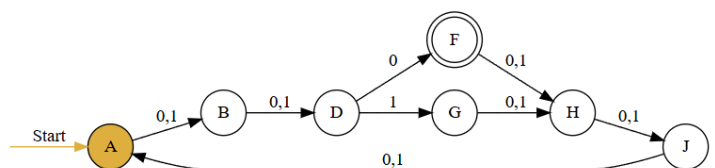
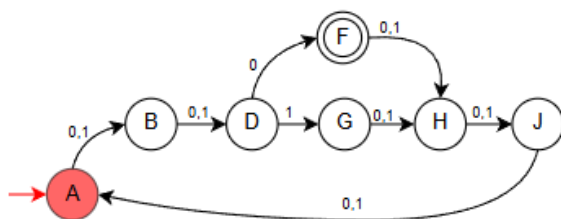
- Nos DFA, faça passo a passo o teste da sequência 101001010, e apresente as imagens de cada passo.

O número binário 101001010 tem 9 dígitos (que é múltiplo de 3 e ímpar) e é de valor par, portanto é aceite por ambos os DFA's.

**DFA 1 para 101001010** Nesta parte confesso que não reparei que dava para fazer na ferramenta UAbALL e fiz tudo no meu DFA, depois apercebi-me que o pedido era mesmo usando a ferramenta, portanto apresento as duas formas, já que o trabalho já tinha sido feito. No meu diagrama a verde representei o estado em que está e a vermelho o estado para o qual transita.

Step-by-Step Input Simulation

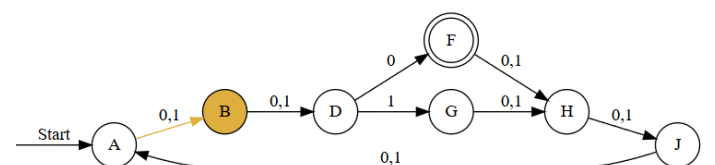
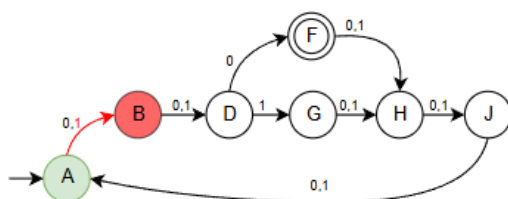
State	To be started
Input	101001010



Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

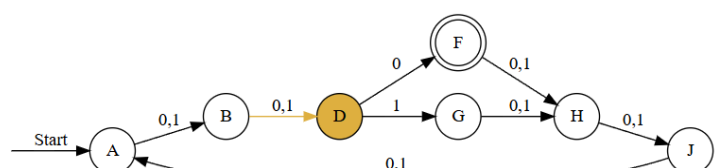
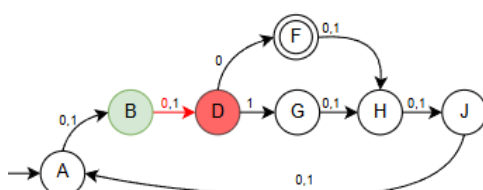
1



Step-by-Step Input Simulation

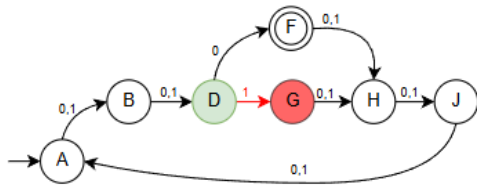
State	In Simulation!
Input	101001010

10

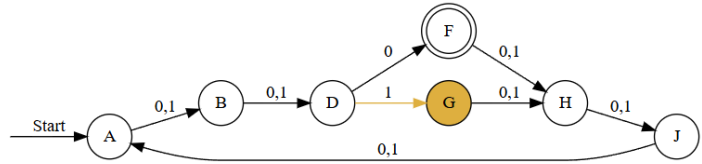


#### Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

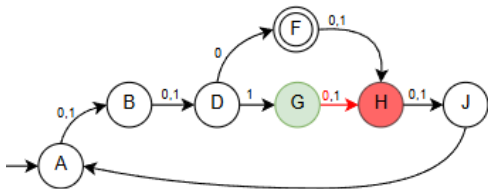


101

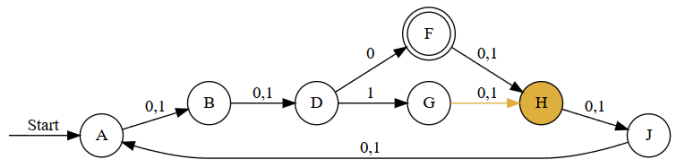


#### Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

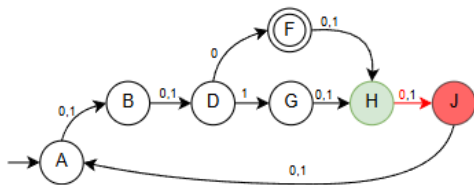


1010

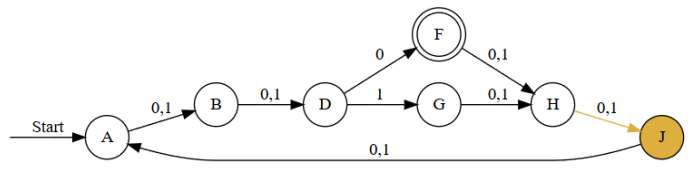


#### Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

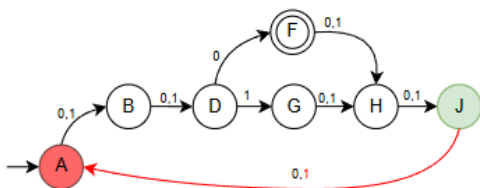


10100

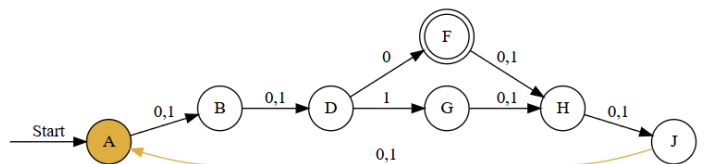


#### Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

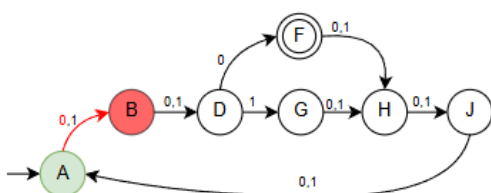


101001

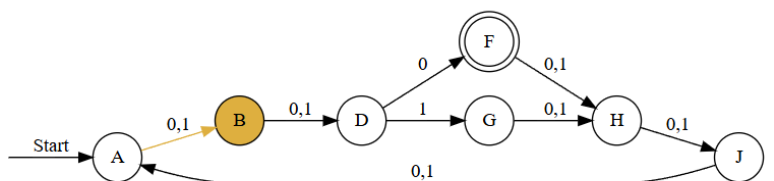


#### Step-by-Step Input Simulation

State	In Simulation!
Input	101001010



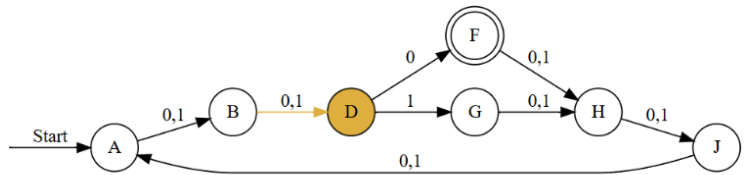
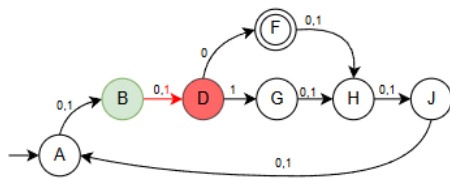
1010010



Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

10100101

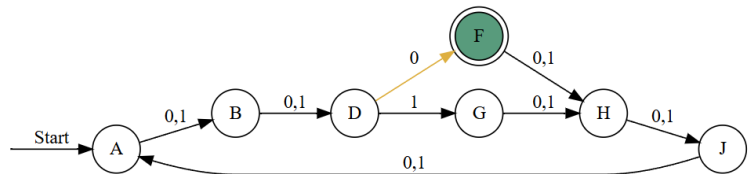
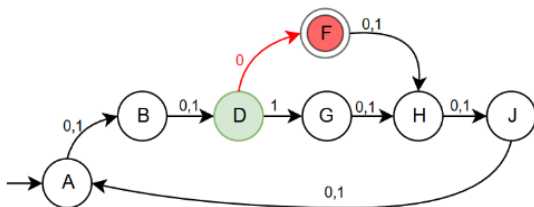


Step-by-Step Input Simulation

The Automaton accepted the entry! Start over or test a new entry

State	Successfull!
Input	101001010

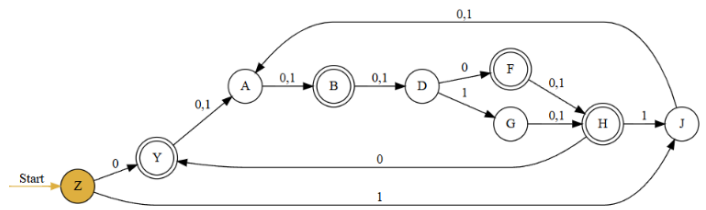
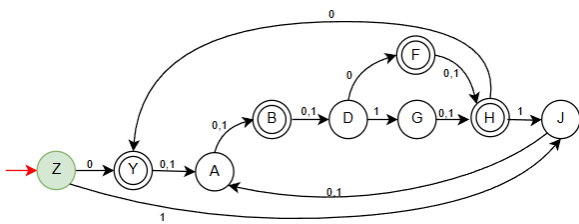
101001010



### DFA 3 para 101001010

Step-by-Step Input Simulation

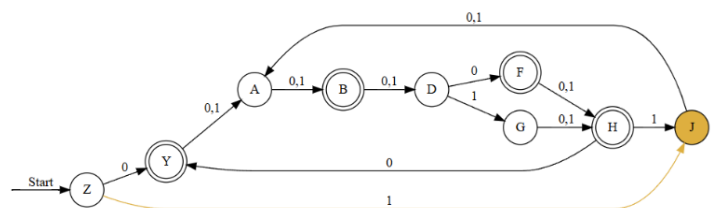
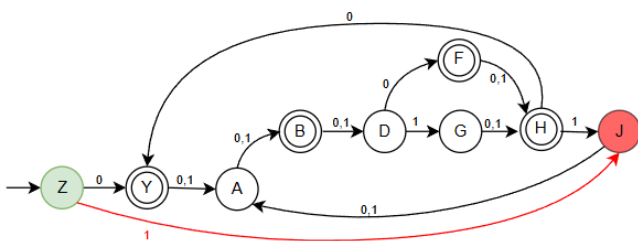
State	To be started
Input	101001010



Step-by-Step Input Simulation

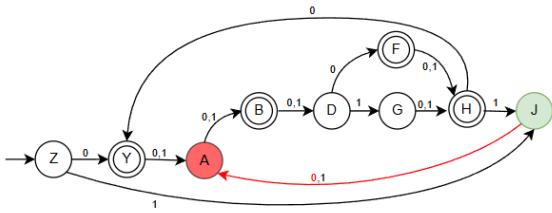
State	In Simulation!
Input	101001010

1

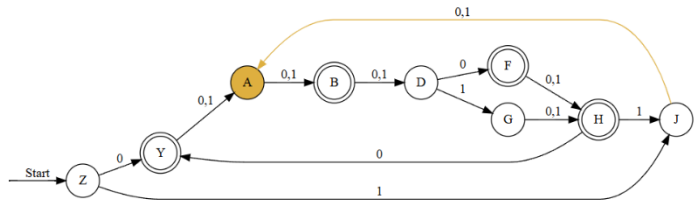


# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

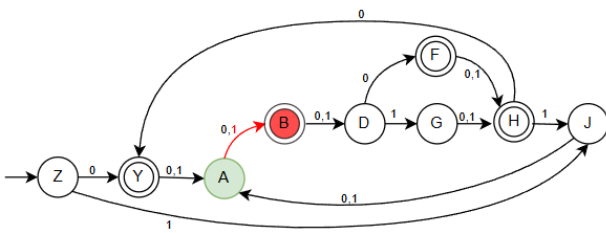


10

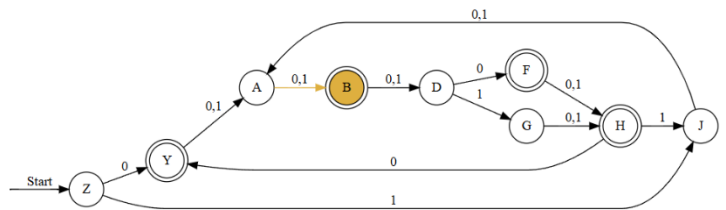


# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

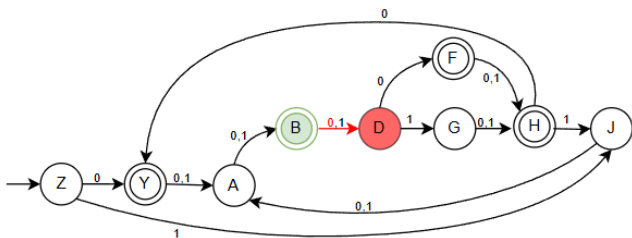


101

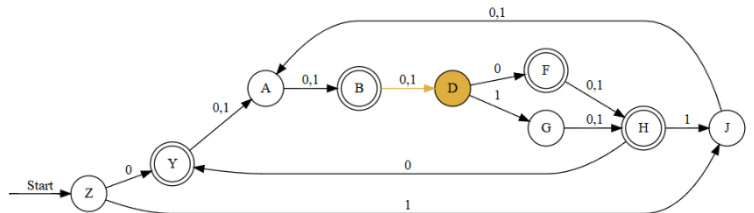


# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

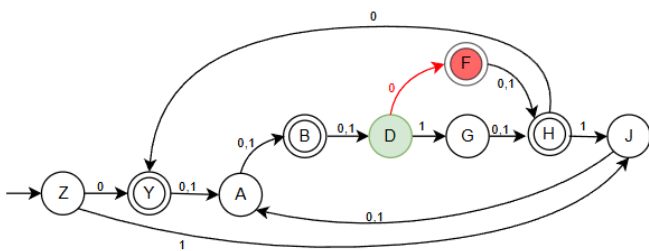


1010

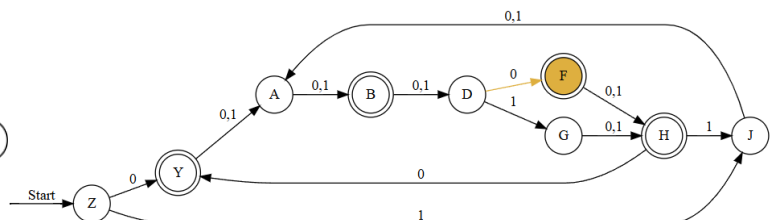


# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010



10100

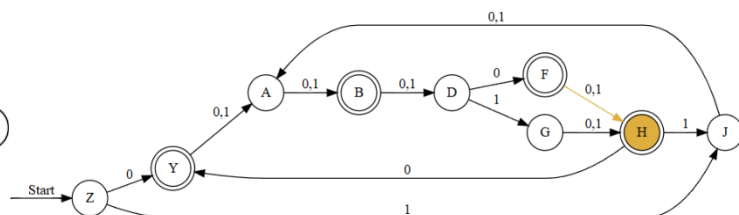
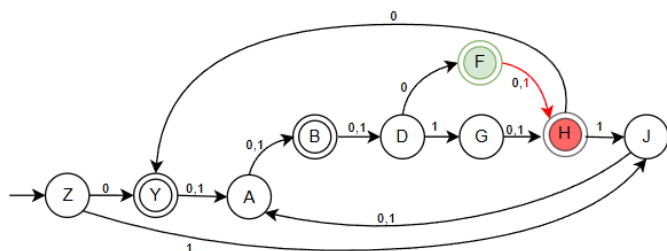




# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

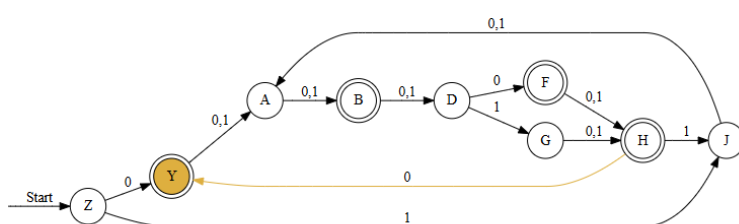
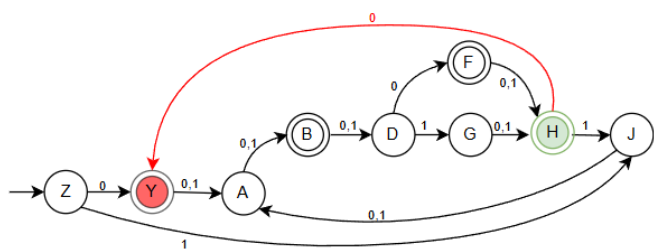
101001



# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

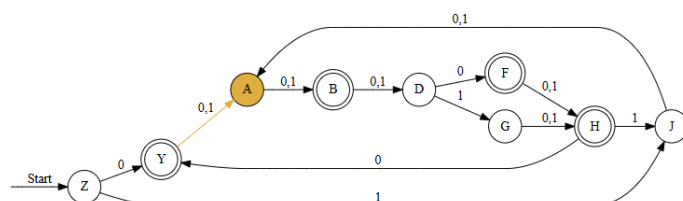
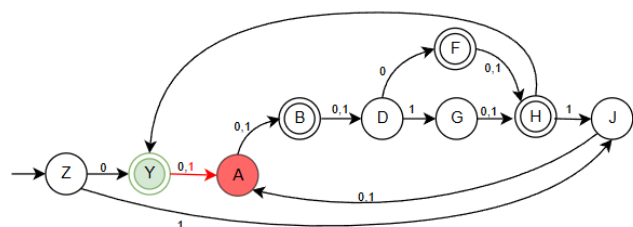
1010010



# Step-by-Step Input Simulation

State	In Simulation!
Input	101001010

10100101

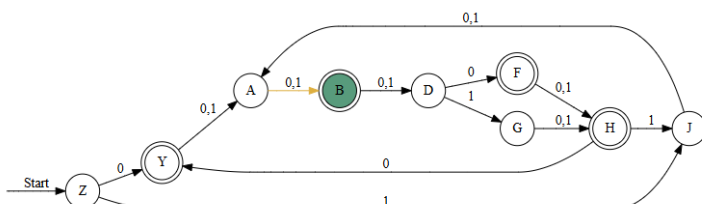
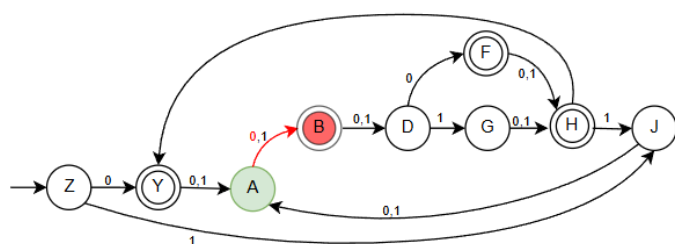


# Step-by-Step Input Simulation

The Automaton accepted the entry! Start over or test a new entry

State	Successfull!
Input	101001010

101001010



De facto, fica comprovado que a sequencia 101001010 é aceite em ambos os autômatos pois em ambos terminam num estado final.

**Bibliografia:**

[https://wiki.dcet.uab.pt/files/index.php/Categoria:Engenharia Inform%C3%A1tica](https://wiki.dcet.uab.pt/files/index.php/Categoria:Engenharia_Inform%C3%A1tica)

[https://www.youtube.com/watch?v=pUYQejZvYPE&ab\\_channel=ComputationalLecturesbyAjayLoura](https://www.youtube.com/watch?v=pUYQejZvYPE&ab_channel=ComputationalLecturesbyAjayLoura)

[https://www.youtube.com/watch?v=HLOAwCCYVxE&ab\\_channel=EasyTheory](https://www.youtube.com/watch?v=HLOAwCCYVxE&ab_channel=EasyTheory)

[https://www.youtube.com/watch?v=wJk82yQhxms&ab\\_channel=International](https://www.youtube.com/watch?v=wJk82yQhxms&ab_channel=International)

[https://www.youtube.com/watch?v=qfVTXwuxEOY&ab\\_channel=MohammadT.Irfan](https://www.youtube.com/watch?v=qfVTXwuxEOY&ab_channel=MohammadT.Irfan)

Hopcroft, Motwani & Ullman. Introduction to Automata Theory, Languages and Computation, 3rd edition. Addison-Wesley. ISBN 0-321-47617-4.

Ferramenta UAbALL: <https://chic.uab.pt/uaball/>

[https://en.wikipedia.org/wiki/Thompson%27s\\_construction](https://en.wikipedia.org/wiki/Thompson%27s_construction)