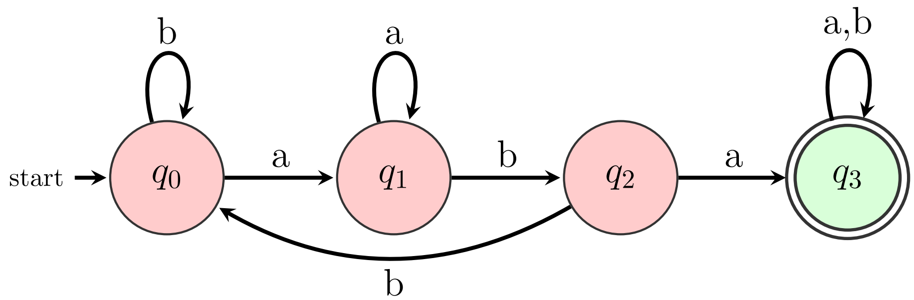
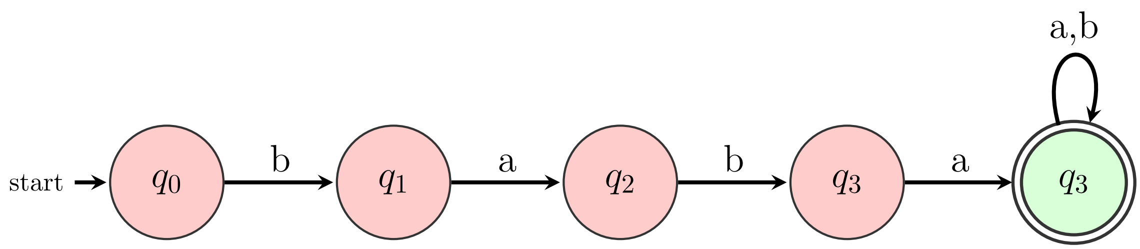
Contents

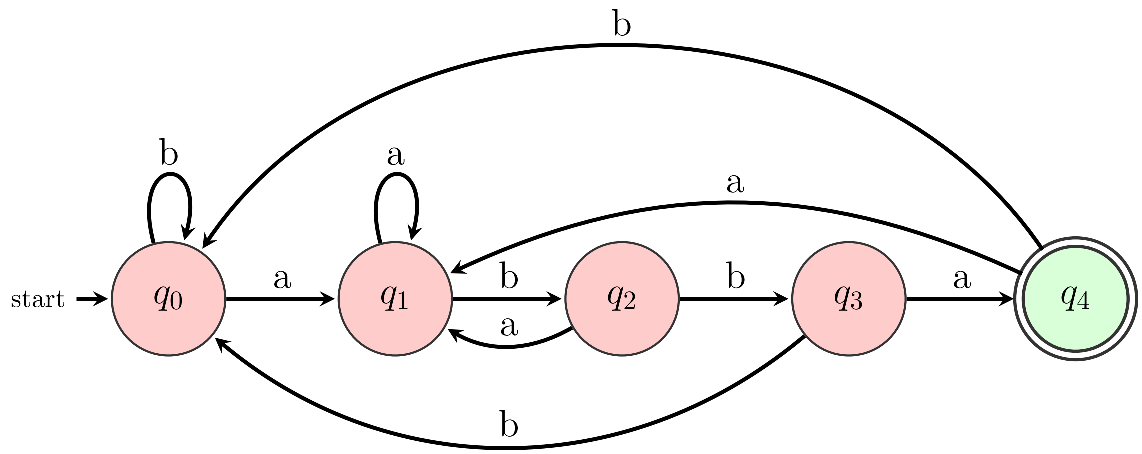
1 DFA to accept strings of a's and b's that contain substring 'aba'



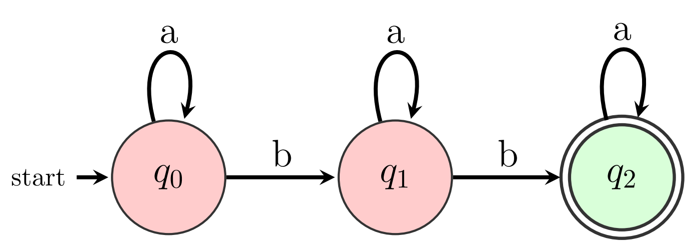
2 To accept strings of a's and b's that start with baba



3 To accept strings of a's and b's that end with abba

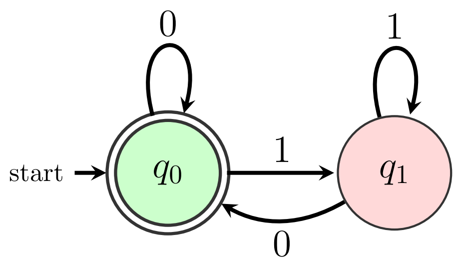


4 To accept strings of a's and b's that contains exactly two b's



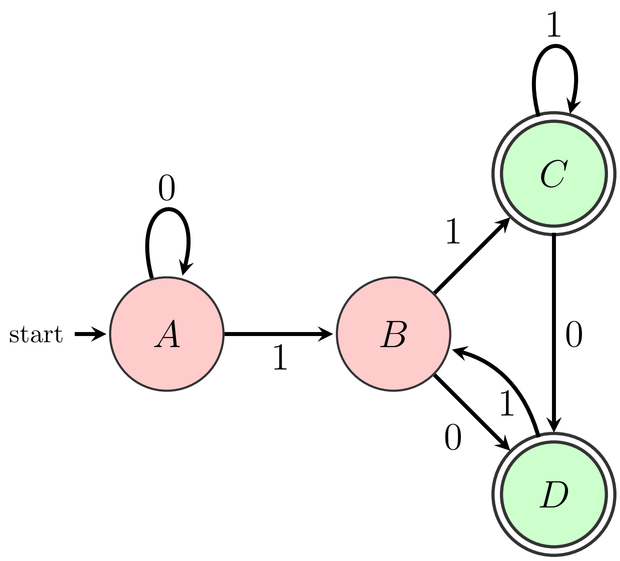
5

L = Construct a DFA which accepts set of all strings over $\Sigma = \{0, 1\}$, which interpreted as a binary number is divisible by 2.



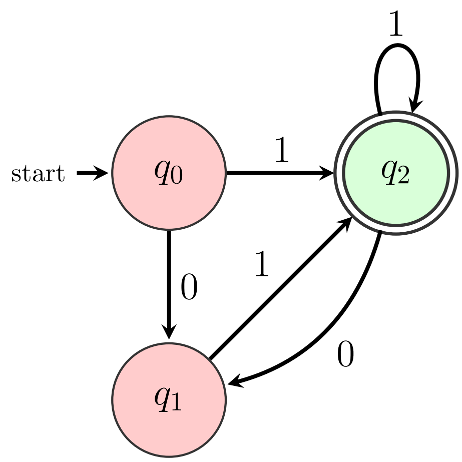
6

L = Strings with next to last symbol 1; where Σ = {0, 1}.

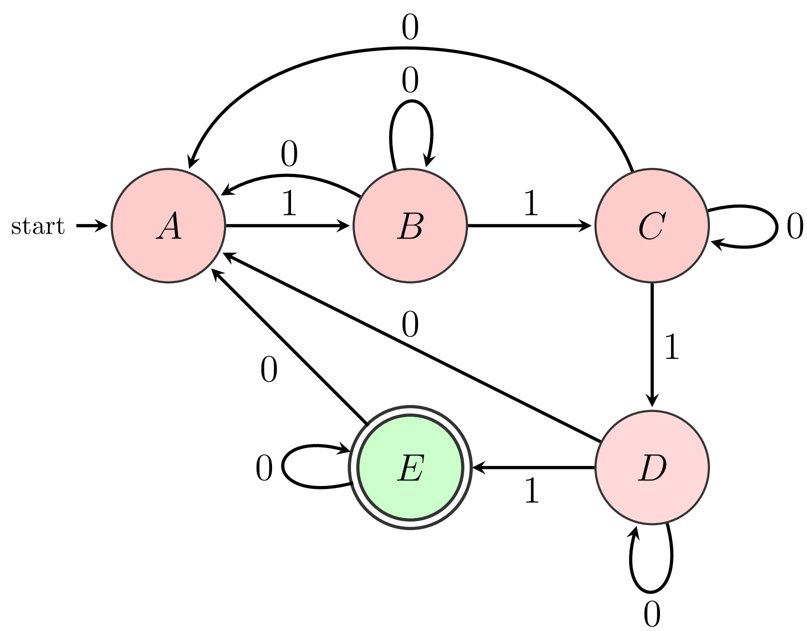


7

L = Strings ending in 1 and not containing 00; where Σ = {0, 1}.



8 With 5 nfa state, construct a dfa whcih lead to worst case state scenario. Final output will show 32 state



8.0.1 Transition Table of the DFA

DFA State	On 0	On 1
{A}	∅	{ B}
∅	∅	∅
{B}	{A, B}	{ C}
{ C}	{A, C}	{ D}
{A, B}	{A, B}	{B, C}
{A, C}	{A, C}	{B, D}
{D}	{A, D}	{E}
{B, C}	{A, B, C}	{C, D}
{B, D}	{A, B, D}	{C, E}
{A, D}	{A, D}	{B, E}
{ E}	{A, E}	{}
{C, D}	{A, C, D}	{D, E}
{A, B, C}	{A, B, C}	{B, C, D}
{B, E}	{A, B, E}	{C}
{A, C, D}	{A, C, D}	{B, D, E}
{D, E}	{A, D, E}	{E}
{B, C, D}	{A, B, C, D}	{C, D, E}
{C, D, E}	{A, C,D, E}	{D, E}
{A, B, C, D}	{A, B, C, D}	{B, C, D, E}
{B, C, D, E}	{A, B, C, D, E}	{C, D, E}
{A, E}	{A, E}	{B}
{A, D, E}	{A, D, E}	{B, E}
{A, C, D, E}	{A, C, D, E}	{B, C, D}
{A,B,C, D, E}	{A, B, C, D, E}	{B,C,D, E}
{A, B, D}	{A, B, D}	{B, C, E}
{C, E}	{A, C, E}	{D}
{B, C, E}	{A, B, C, E}	{C, D}
{A, C, E}	{A, C, E}	{B, D}
{A, B, C, E}	{A, B, C, E}	{B, C, D}
{A, B, E}	{A, B, E}	{B, C}
{B, D, E}	{A, B, D, E}	{C, E}
{A, B, D, E}	{A, B, D, E}	{B, C, E}

Table 1: DFA Transition Table

8.1 Suset construction

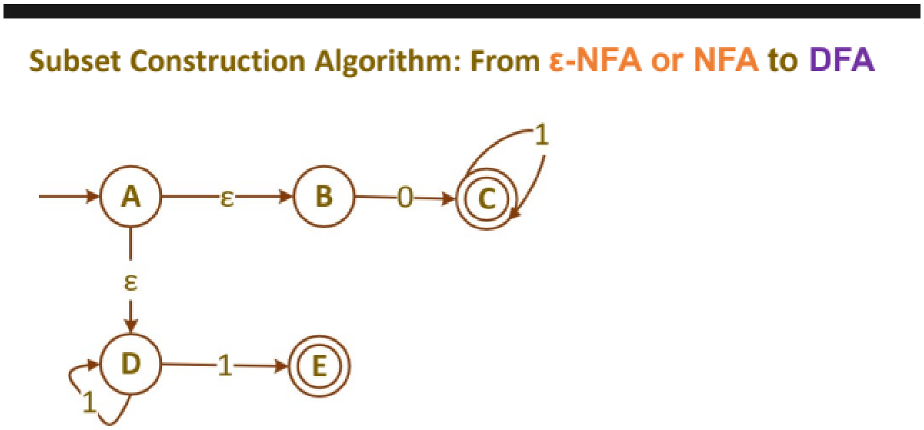


Figure 1: Caption

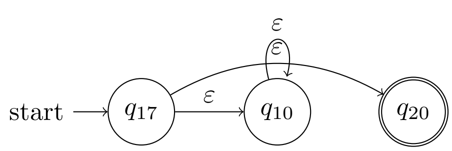
State	**0**	**1**
$\{A, B, D\}$ (start)	$\{C\}$	$\{D, E\}$
$\{C\}$ (accept)	$\emptyset$	$\{C\}$
$\{D, E\}$ (accept)	$\emptyset$	$\{D, E\}$
$\emptyset$	$\emptyset$	$\emptyset$

Table 2: DFA Transition Table

9 Regular expression to FA

9.1 (((0+10)*1)*01*0)*

From	Input	To
q_{17}	ε	q_{10}, q_{20}
q_{10}	ε	q_7, q_{11}
q_7	ε	q_1, q_8
q_1	ε	q_2, q_4
q_2	0	q_3
q_3	ε	q_6
q_4	1	q_5
q_5	0	q_6
q_6	ε	q_1, q_8
q_8	1	q_9
q_9	ε	q_7, q_{11}
q_{11}	0	q_{12}
q_{12}	ε	q_{13}
q_{13}	ε	q_{14}
q_{13}	1	q_{15}
q_{15}	ε	q_{13}
q_{14}	0	q_{16}
q_{16}	ε	q_{10}, q_{20}



10 FA to RE

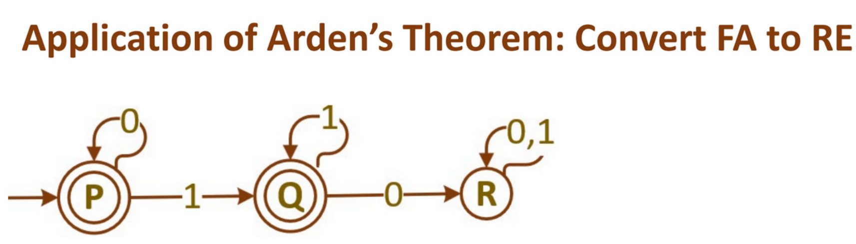


Figure 2: Caption

P = 0*
Q = 0*11*

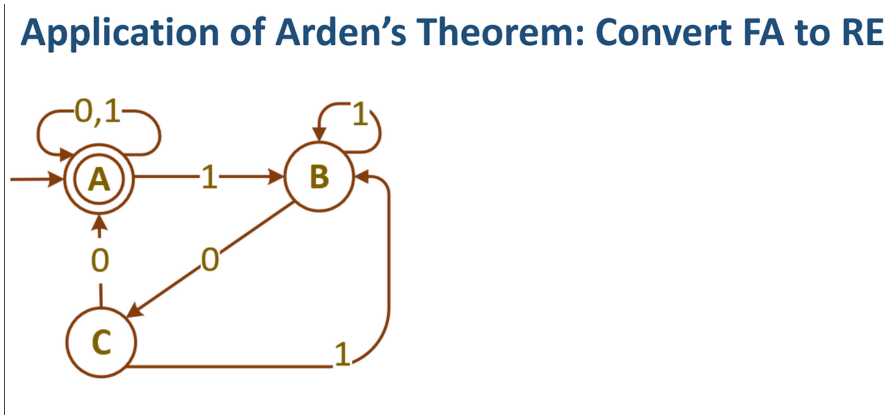


Figure 3: Caption

A = (0 + 1 + (1 (1 + 01)* 00)*

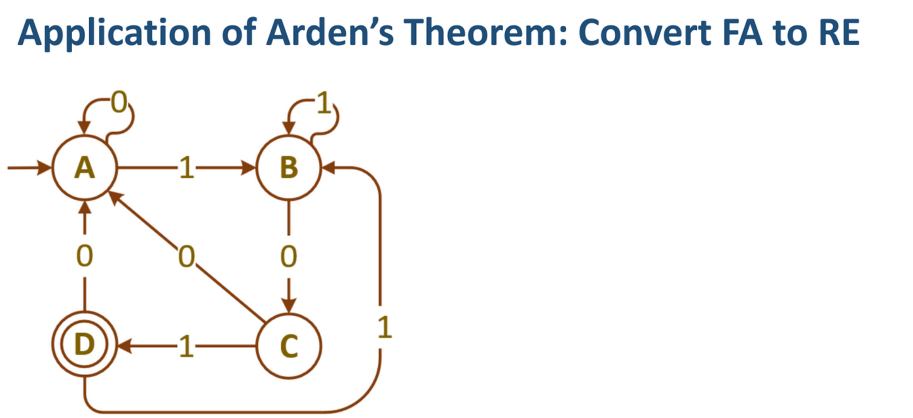
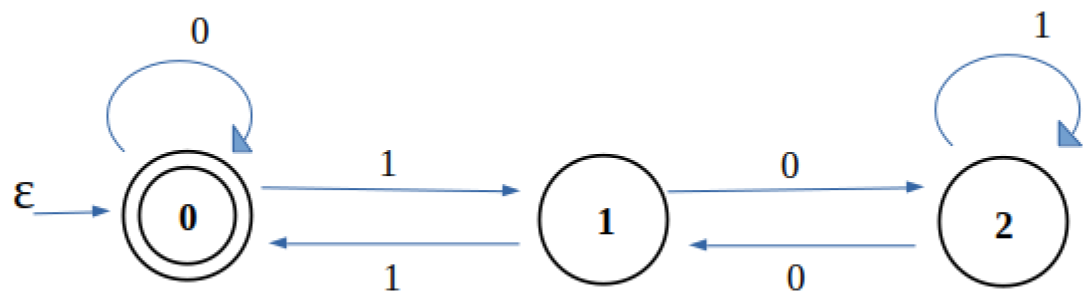


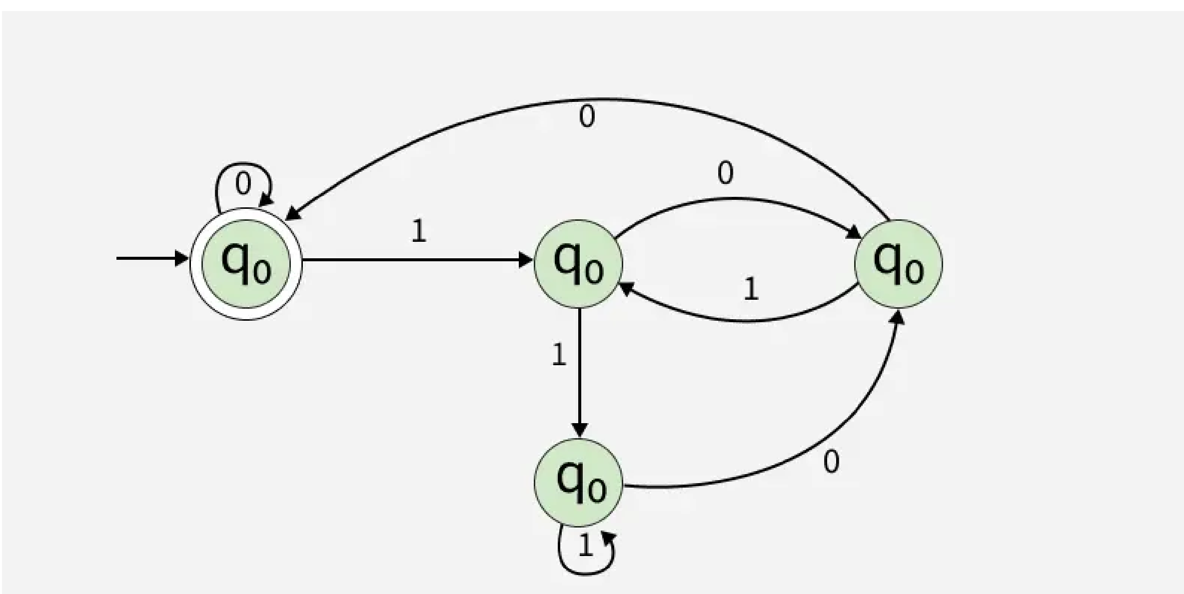
Figure 4: Caption

A = (0 + 1 (1 + 011)* (00+010))*
D =

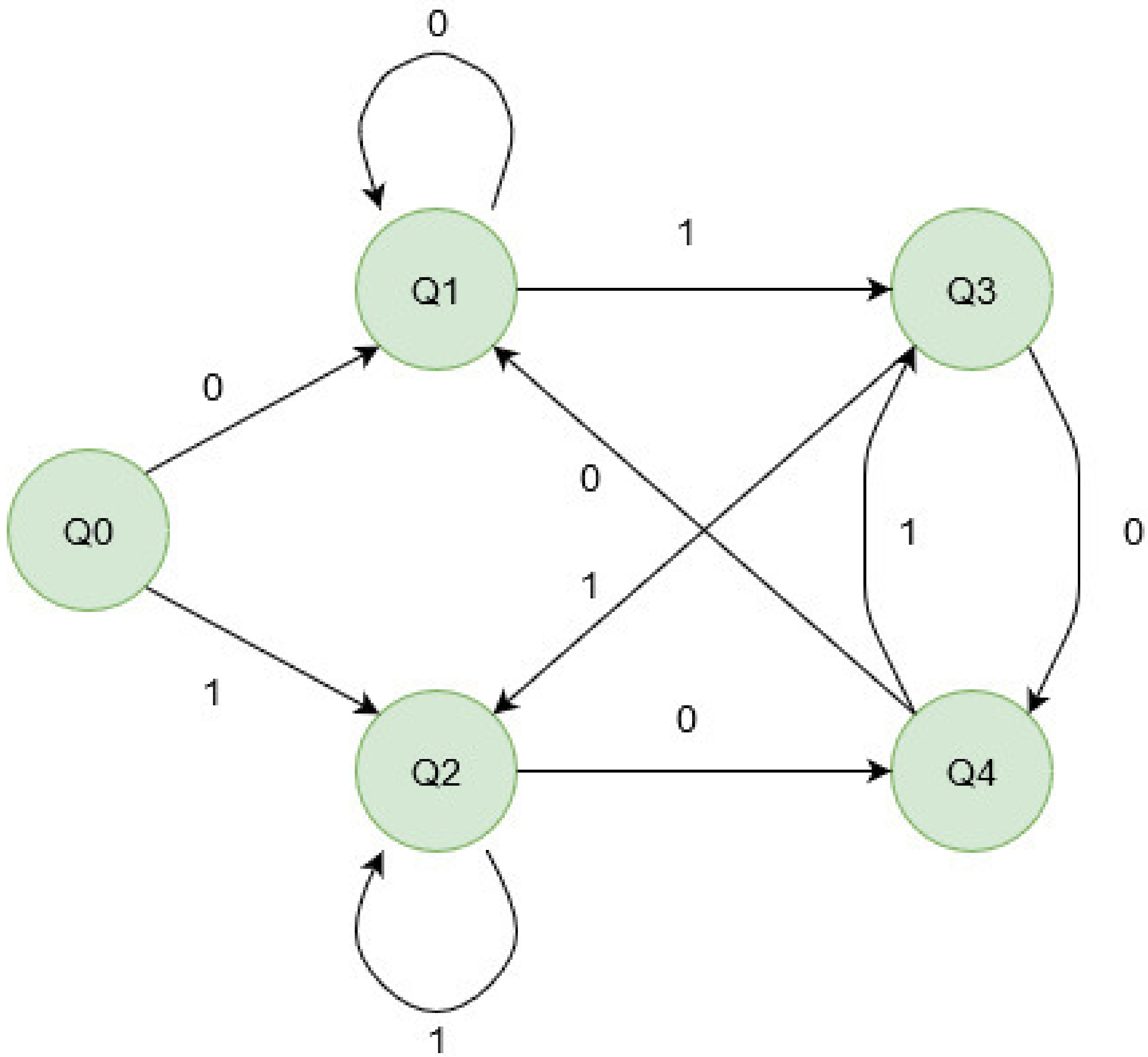
11 **Construct DFA, which accepts set of all strings over 0, 1 which interpreted as binary number is divisible by 3.**



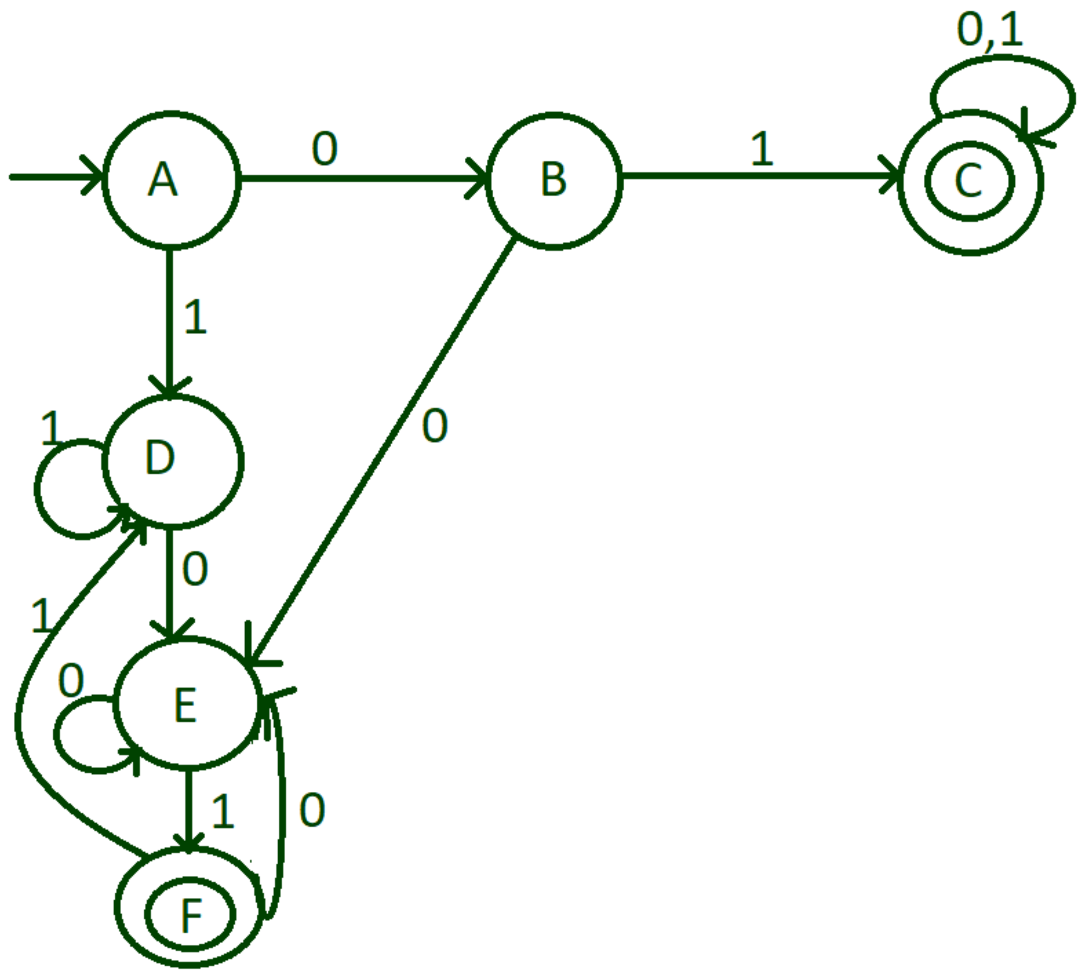
12 **Construct DFA, which accepts set of all strings over 0, 1 which interpreted as binary number is divisible by 4.**



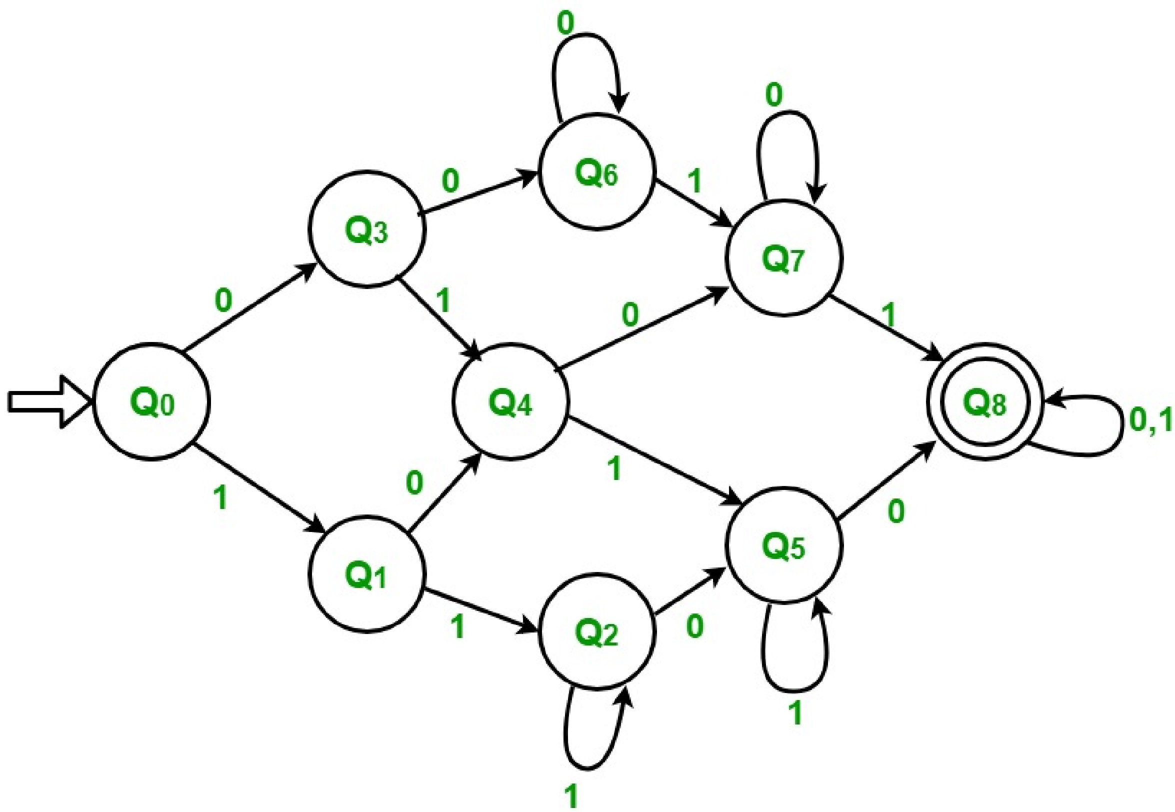
13 DFA that checks if a string ends with "01" or "10"



14 DFA to accept Binary strings that starts or ends with "01"



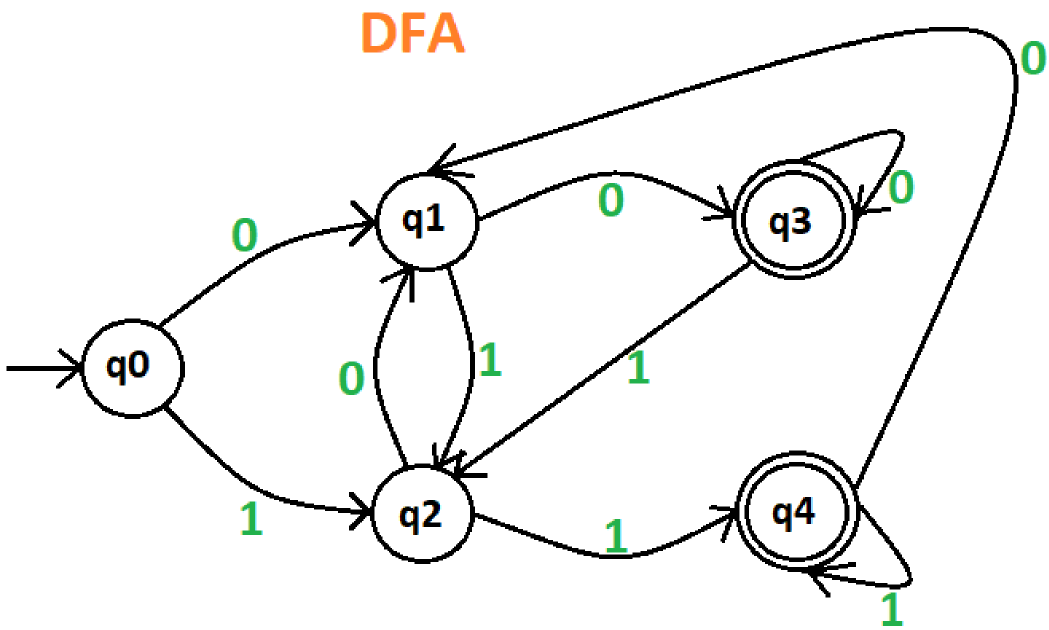
15 DFA of a string with at least two 0's and at least two 1's



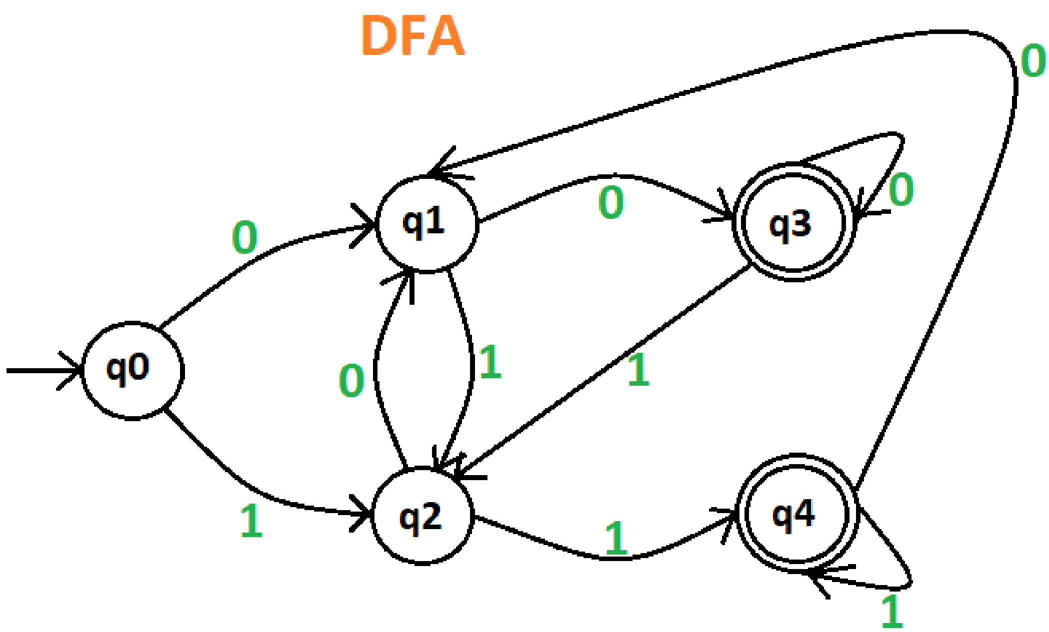
15.1 RE

$$
\begin{aligned}
&(0+1)^*0(0+1)^*0(0+1)^*1(0+1)^*1(0+1)^* \\
&\quad + \ (0+1)^*0(0+1)^*1(0+1)^*0(0+1)^*1(0+1)^* \\
&\quad + \ (0+1)^*0(0+1)^*1(0+1)^*1(0+1)^*0(0+1)^* \\
&\quad + \ (0+1)^*1(0+1)^*0(0+1)^*0(0+1)^*1(0+1)^* \\
&\quad + \ (0+1)^*1(0+1)^*0(0+1)^*1(0+1)^*0(0+1)^* \\
&\quad + \ (0+1)^*1(0+1)^*1(0+1)^*0(0+1)^*0(0+1)^*
\end{aligned}
$$

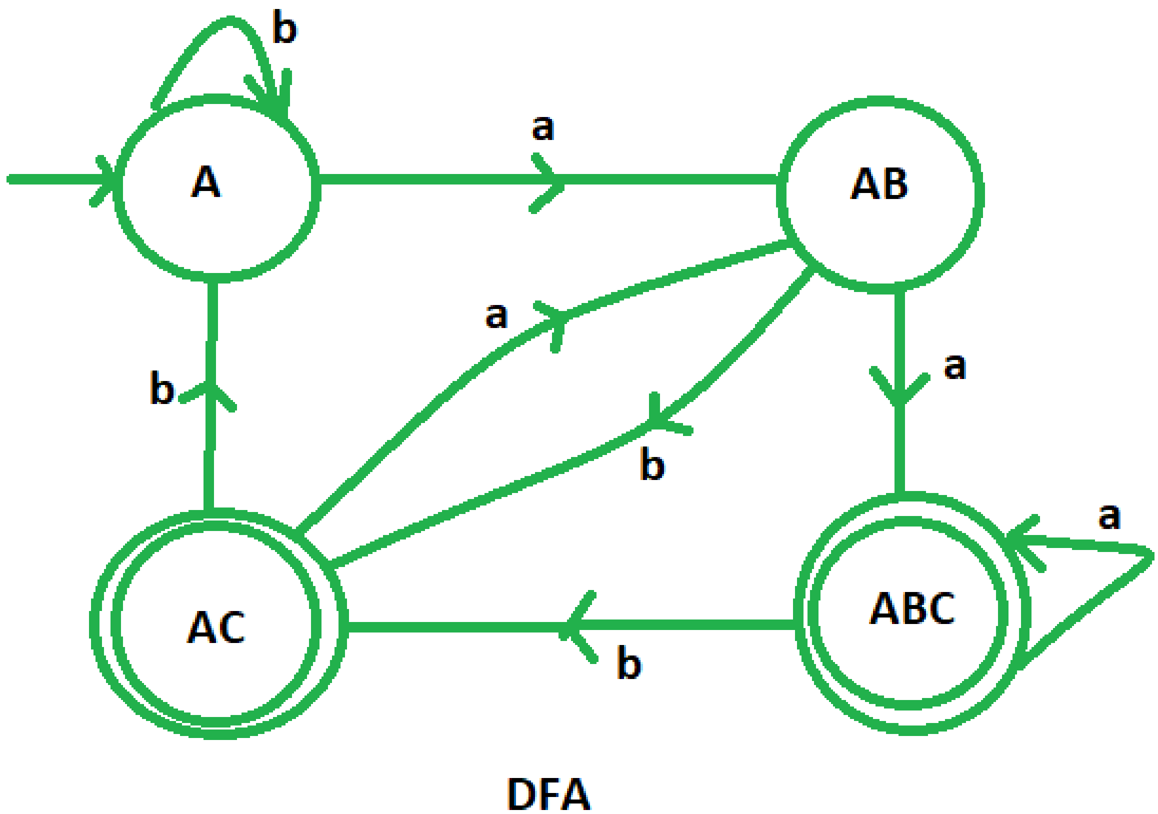
16 accept 00 and 11 at the end of a string

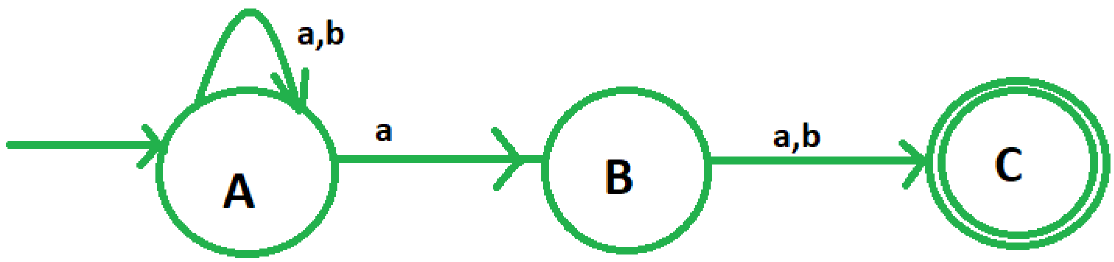


17 DFA for accepting the language $L = \{a^n b^m \mid n + m \text{ is even}\}$



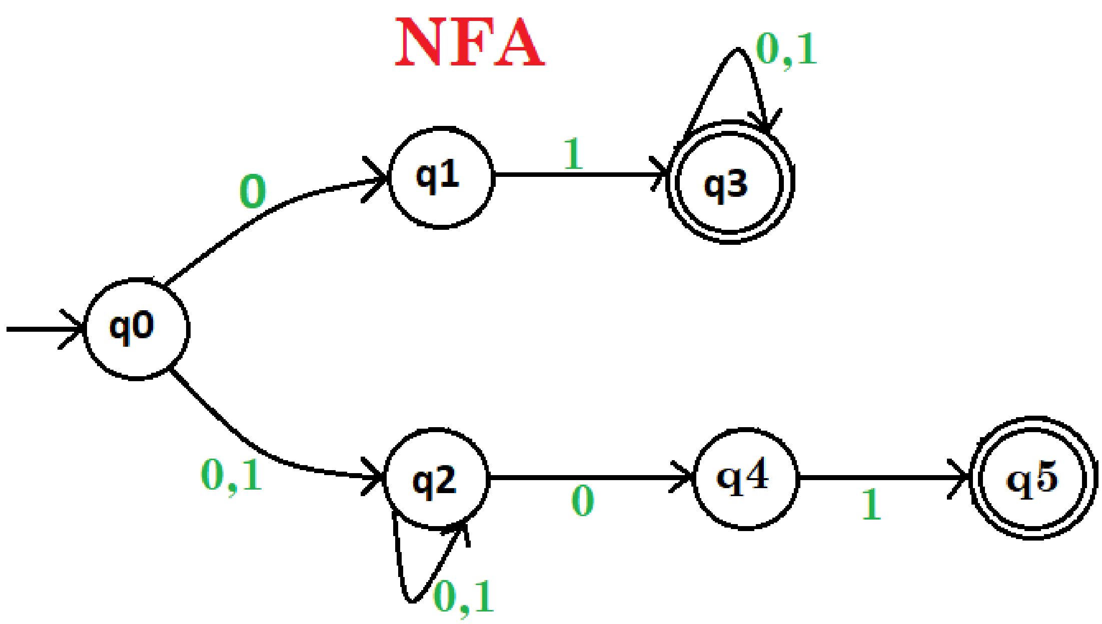
18 DFA of a string in which 2nd symbol from RHS is 'a'

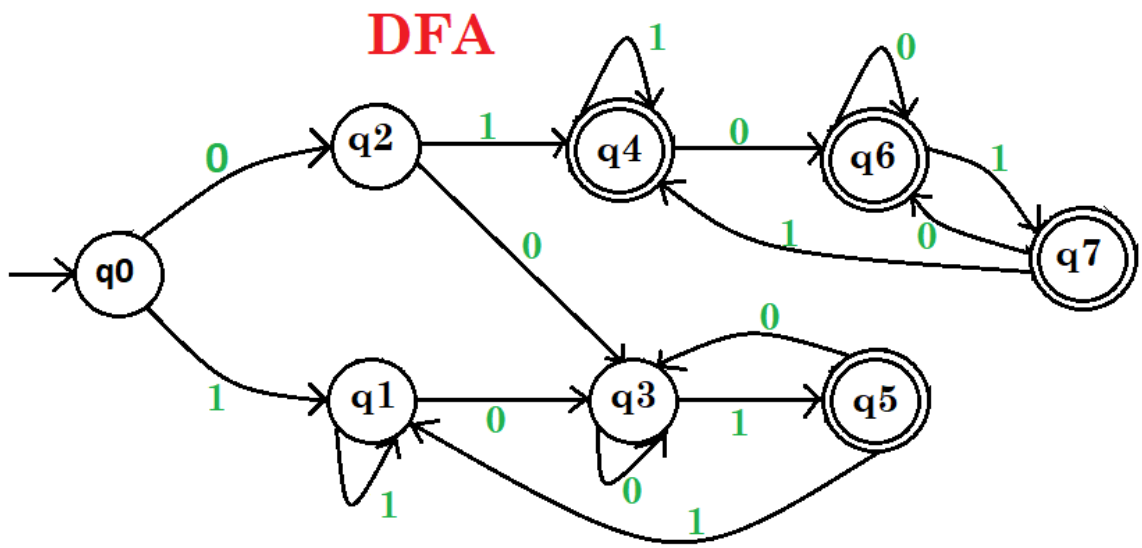




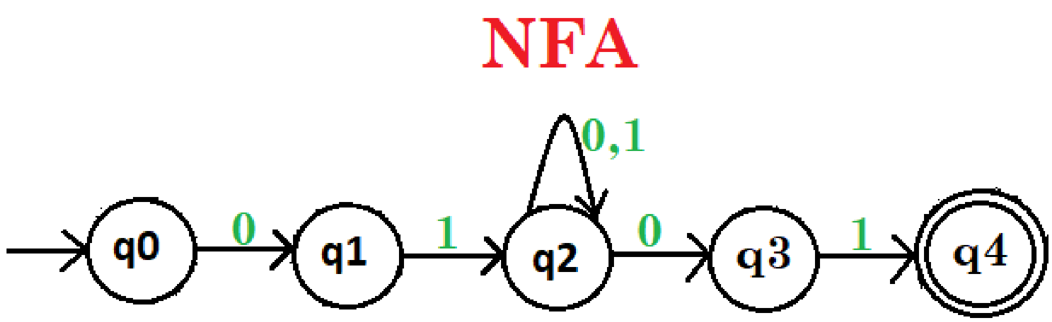
NFA

19 **Draw a deterministic and non-deterministic finite automate which either starts with 01 or end with 01 of a string containing 0, 1 in it, e.g., 01010100 but not 000111010.**

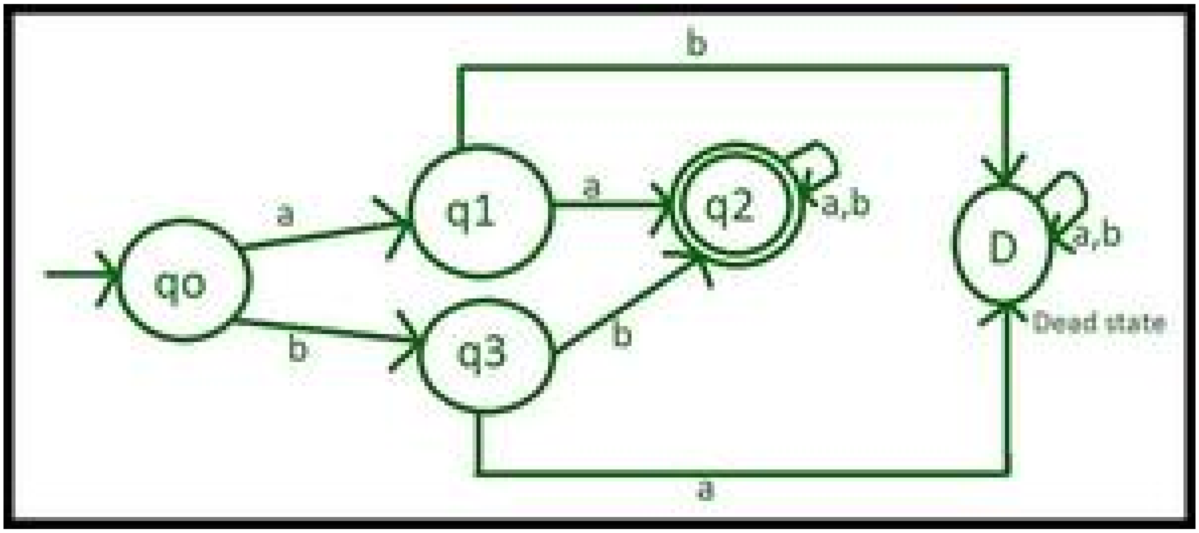




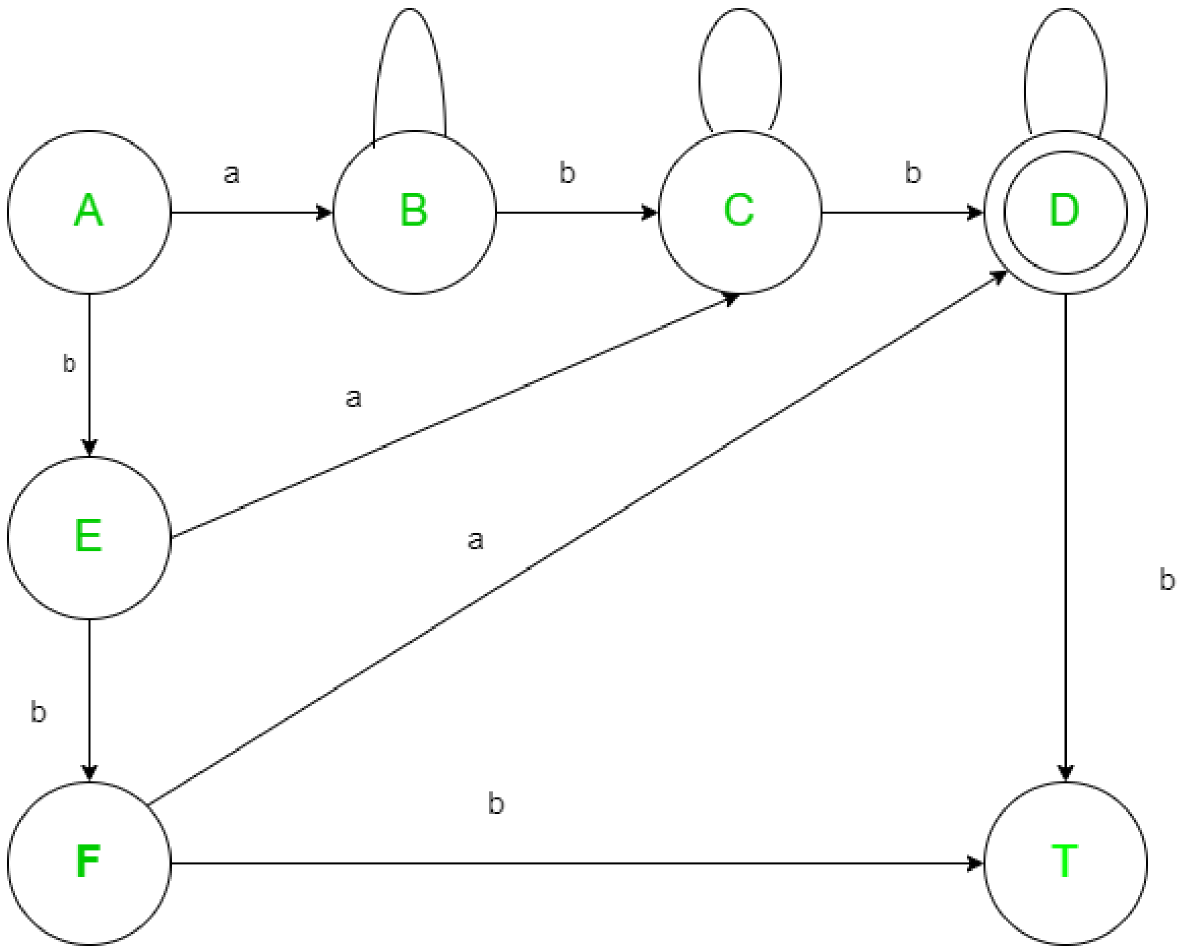
20 **Draw a non-deterministic finite automate which starts with 01 and ends with 01 of a string containing 0, 1 in it, e.g., 01000101 but not 000111001.**



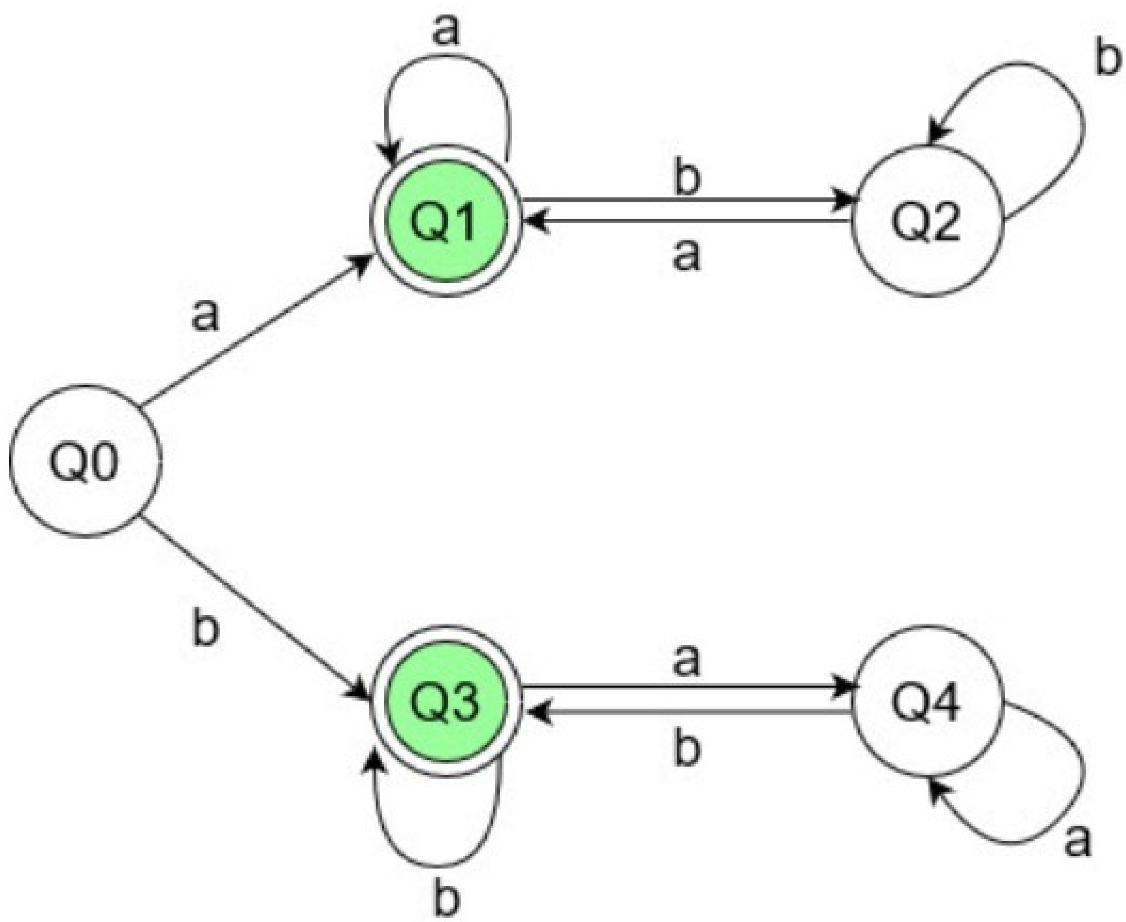
21 Construct a DFA that Start With aa or bb



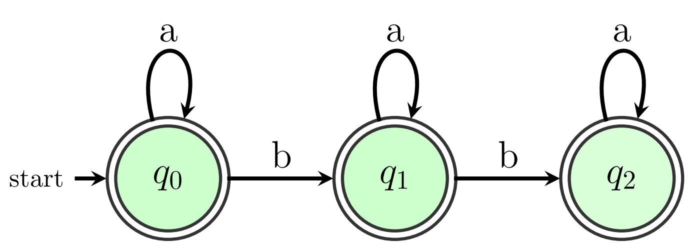
22 DFA that Accepts All the Strings With At Least 1 'a' and Exactly 2 b's



23 Program to build a DFA to accept strings that start and end with same character



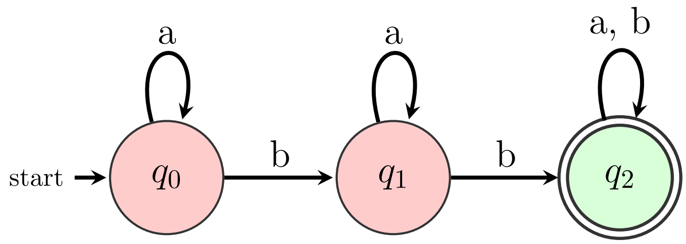
24 at most two b



24.1 RE

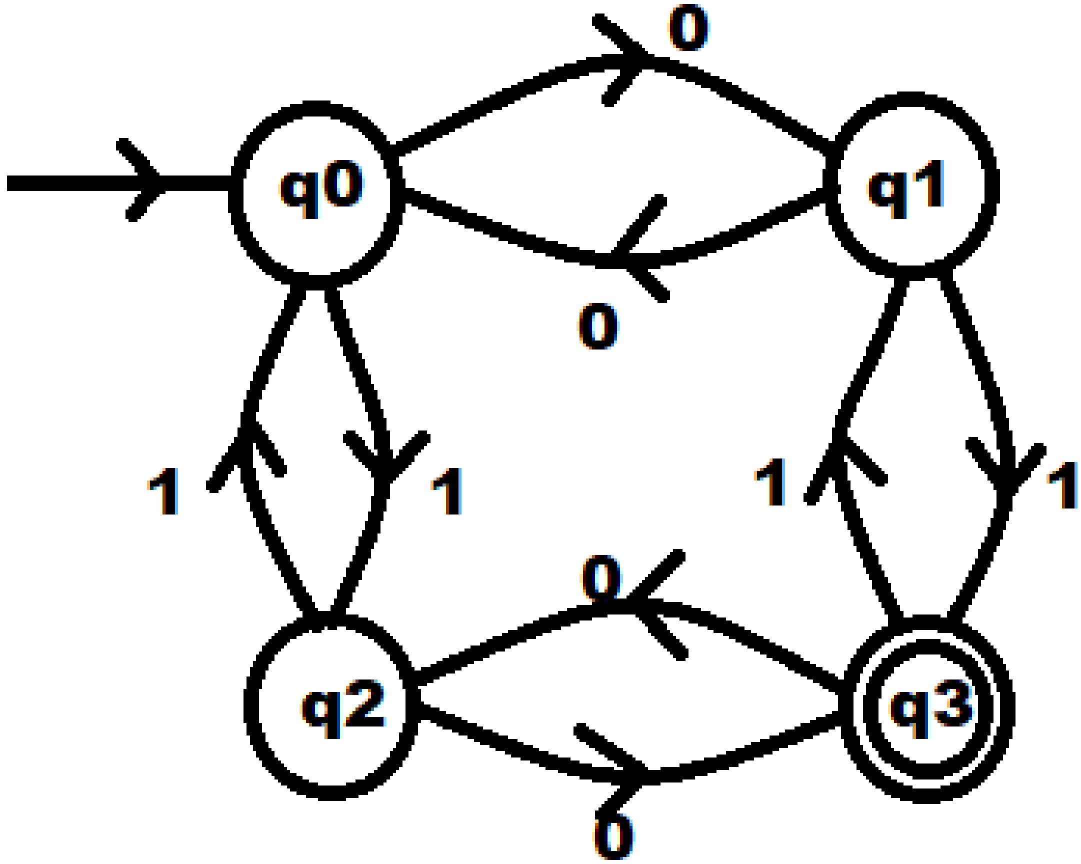
$$\mathbf{a}^* \ + \ \mathbf{a}^*\mathbf{b}\mathbf{a}^* \ + \ \mathbf{a}^*\mathbf{b}\mathbf{a}^*\mathbf{b}\mathbf{a}^*$$

25 at least 2 b



$$\mathbf{a}^*\mathbf{b}\mathbf{a}^*\mathbf{b}(\mathbf{a}+\mathbf{b})^*$$

26 DFA accepting odd number of 0s and odd number of 1s

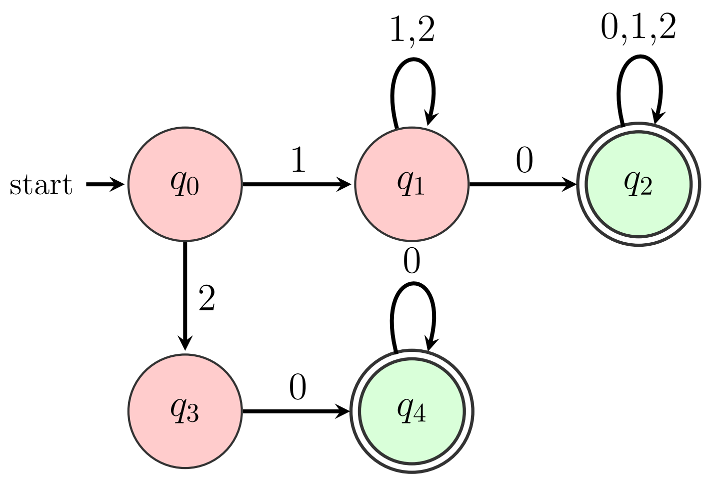


27 Incourse - 27

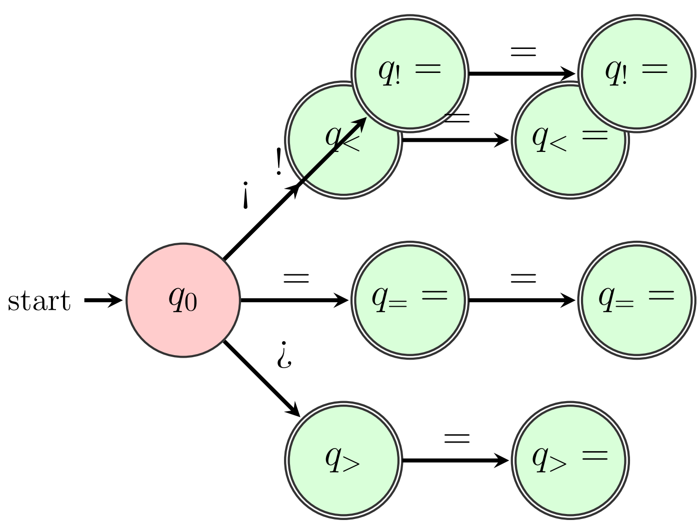
27.1 DFA and Regular Expression over the Alphabet $\{0, 1, 2\}$

A Deterministic Finite Automaton (DFA) and regular expression are to be constructed over the alphabet $\{0, 1, 2\}$ to accept only those strings that satisfy the following constraints:

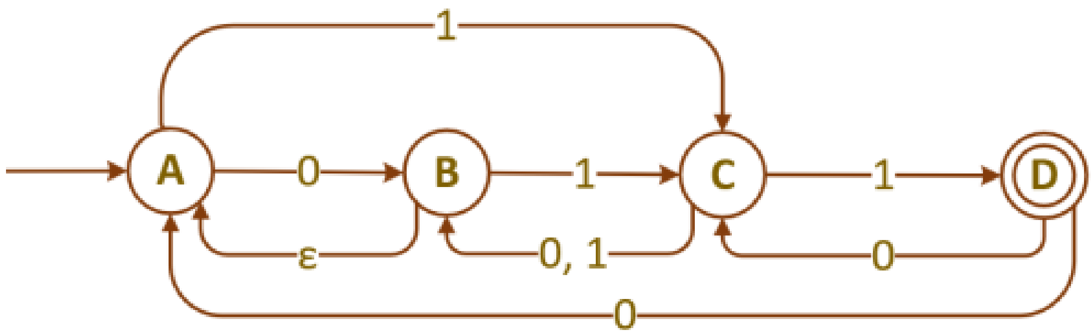
- If the string starts with 1, it must contain at least one occurrence of 0 (there is no restriction on the number of 1s or 2s in the string).
- If the string starts with 2, it must contain exactly one occurrence of 2, and the second-to-last symbol of the string must always be 0.



27.2 DFA for recognizing relational operators



27.3 Convert the given NFA with ε-transitions into an equivalent Deterministic Finite Automaton (DFA). What would be the worst-case outcome of such a transformation in terms of the number of nodes?



27.3.1 Transition Table of the DFA

DFA State	On 0	On 1
{A}	{A,B}	{C}
{A,B}	{A,B}	{C}
{C}	{A,B}	{A,B,D}
{A,B,D}	{A,B,C}	{C}
{A,B,C}	{A,B}	{A,B,C,D}
{A,B,C,D}	{A,B,C}	{A,B,C,D}

Table 3: DFA Transition Table

Start state: $\{A\}$
Accepting states: $\{A, B, D\}, \{A, B, C, D\}$

27.3.2 FA to RE

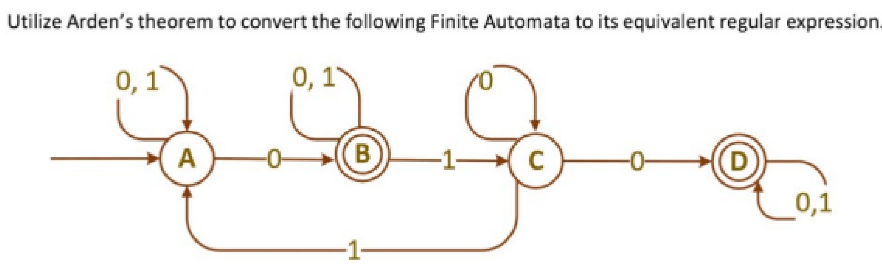
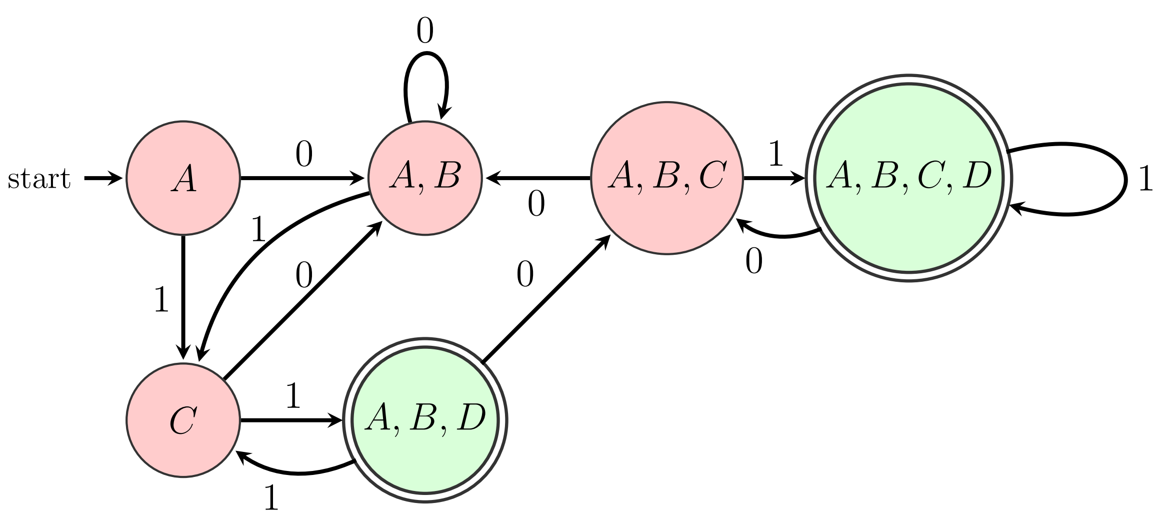


Figure 5: Caption

A = ((0+1) + 0 (0+1)*10*1)*

27.3.3 DFA Diagram

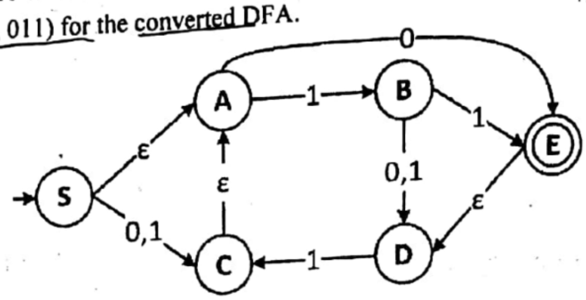


27.3.4 Worst-case Outcome

For an NFA with n states, the subset-construction may produce a DFA with up to 2^n states. Here $n = 4$, so the worst-case DFA has $2^4 = 16$ states. In this particular case, only 6 DFA states are reachable.

27.4 Final 27 (1b)

(b) Produce the output of the following functions: ε-closure (δ (S, 0)), and δ(B, 1). Convert the given NFA with ε-transitions into an equivalent Deterministic Finite Automaton (DFA). Finally, calculate $\hat{\delta}(q_0, 011)$ for the converted DFA.



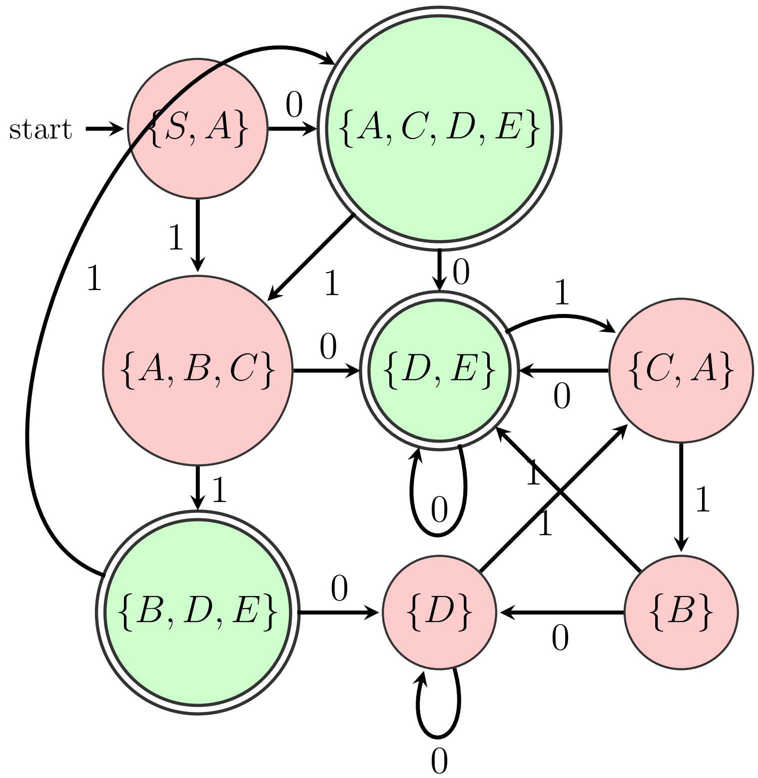
27.4.1 Transition Table of the DFA

DFA State	On 0	On 1
{S, A}	{A,C, D, E}	{A, B, C}
{A,C, D, E}	{D, E}	{A, B, C}
{A, B, C}	{D, E}	{B, E, D}
{D, E}	{}	{C, A}
{B, D, E}	{D}	{A,C,D, E}
{C, A}	{D, E}	{B}
{D}	{}	{C, A}
{B}	{D}	{D, E}

Table 4: DFA Transition Table

Start state: $\{S, A\}$
Accepting states: $\{A, C, D, E\}, \{D, E\}, \{B, E, D\}$

27.5 DFA



27.6 Final 2 a

2 (a) Convert the following Moore Machine to its equivalent Mealy Machine. Determine the output for the input string %@#@ for both the Moore Machine and its equivalent Mealy Machine.

Present State	Transition Function – δ			Output λ
	Next state for Input: #	Next state for Input: @	Next state for Input: %	
→A	B	A	E	$
B	B	C	D	£
C	E	E	C	$
D	A	D	C	&
E	D	B	A	*

Present State	Input: #		Input: @		Input: %	
	State	Output	State	Output	State	Output
$\rightarrow A$	B	£	A	$	E	*
B	B	£	C	$	D	&
C	E	*	E	*	C	$
D	A	$	D	&	C	$
E	D	&	B	£	A	$

Outputs for input string %@#@

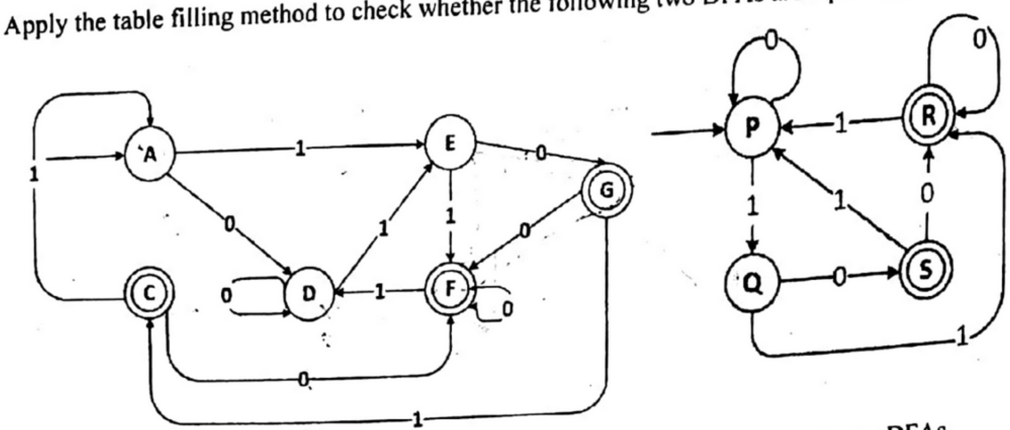
For Moore Machine:

$$\$ * \pounds\pounds\$$$

For Mealy Machine:

$$*\pounds\pounds\$$$

(b) Apply the table filling method to check whether the following two DFAs are equivalent or not.



Explain why this algorithm is effective in correctly identifying equivalence between DFAs.

DFA State	P1	P2	P3	P4	P5	P6
S0	A, D, E, P, Q	C, F, G, R, S				
S0	A, D P	C, F, R	E, Q	G, S		
S0	A, P	C, F, R	E, Q	G	D	S

Table 5: Equivalence Test

3 (a) Construct a regular expression over the alphabet {0,1,2} that accepts strings with at most two occurrences of 0, with no restrictions on the input 1 and 2.

Use Pumping Lemma to show whether the language L is regular or not.

$$L = \{a^{i^2} \mid i \geq 1\}$$

$$(1+2)^* \;+\; (1+2)^*\,0\,(1+2)^* \;+\; (1+2)^*\,0\,(1+2)^*\,0\,(1+2)^*$$

27.6.1 FA to RE

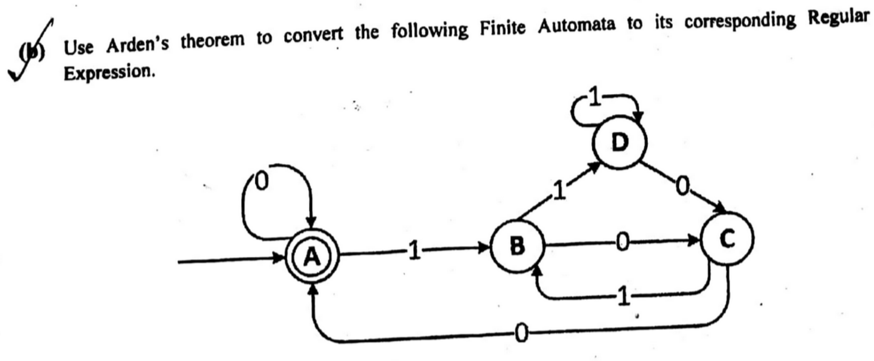
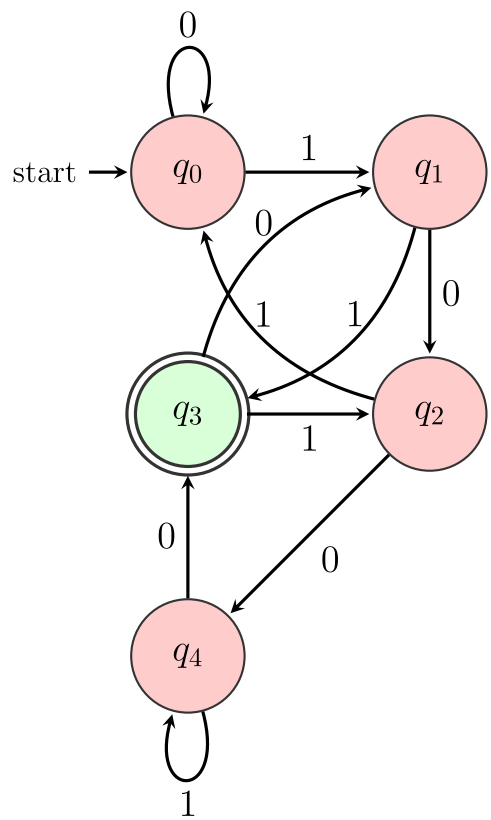


Figure 6: Caption

$A = ((0+1((0+11^*0)1)^*(0+11^*0)0)$

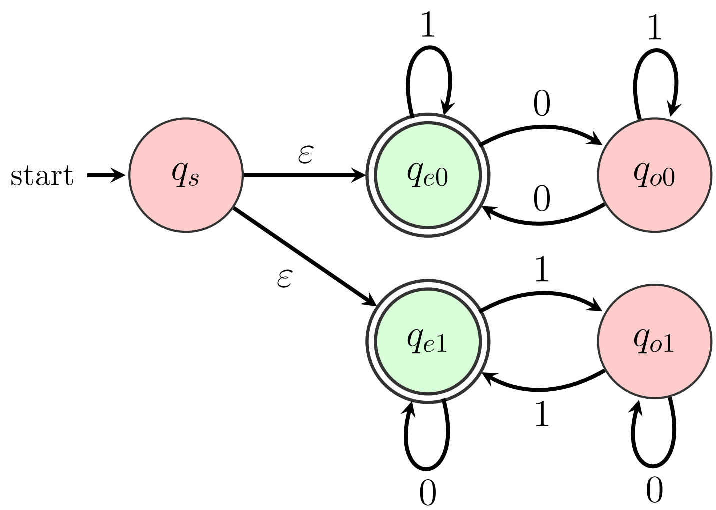
28 26 Batch

28.1 DFA that will accept all binary numbers starting with 0 and remainder is always 11 when divisor is 101 (Final 7a)

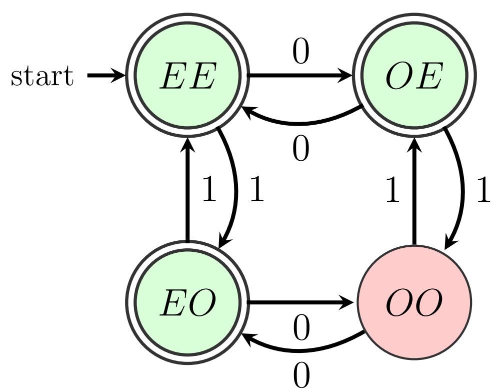


29 25 batch

29.1 (i) ε-NFA for strings over $\{0, 1\}$ with either an even #0s or an even #1s

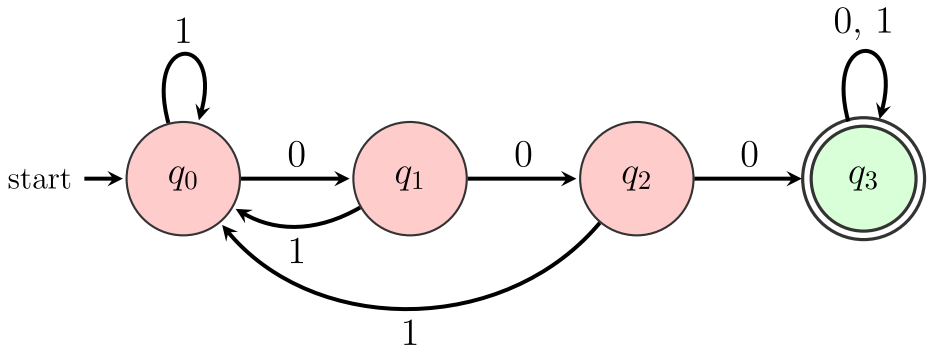


29.2 (ii) Equivalent DFA (subset construction, minimized)



30 24 Batch

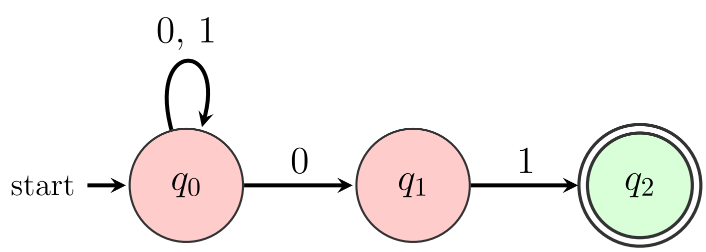
30.1 Three consecutive zeros, not necessarily at the end



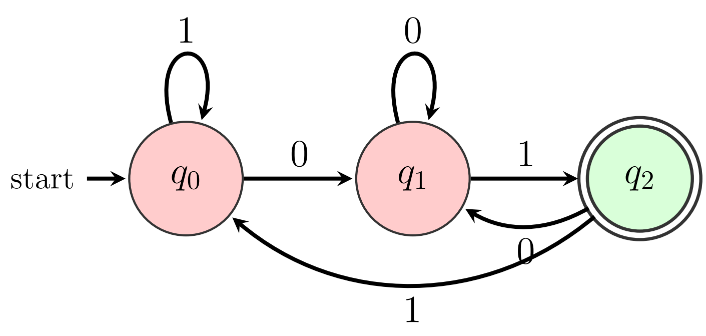
$$(\mathbf{0}+\mathbf{1})^*\,\mathbf{000}\,(\mathbf{0}+\mathbf{1})^*$$

31 23 Batch

31.1 Ends with 01

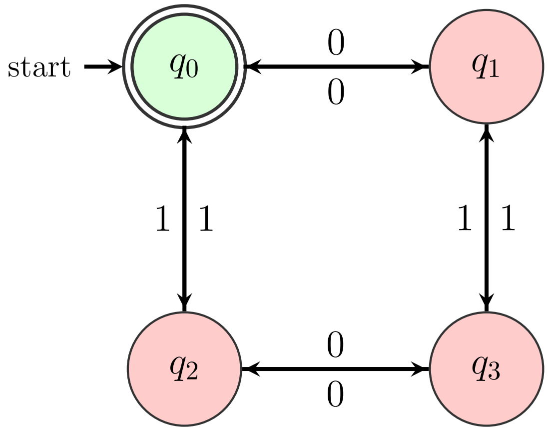


NFA

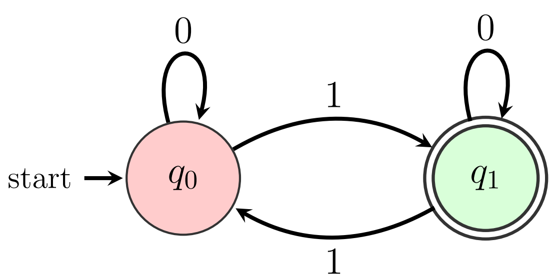


DFA

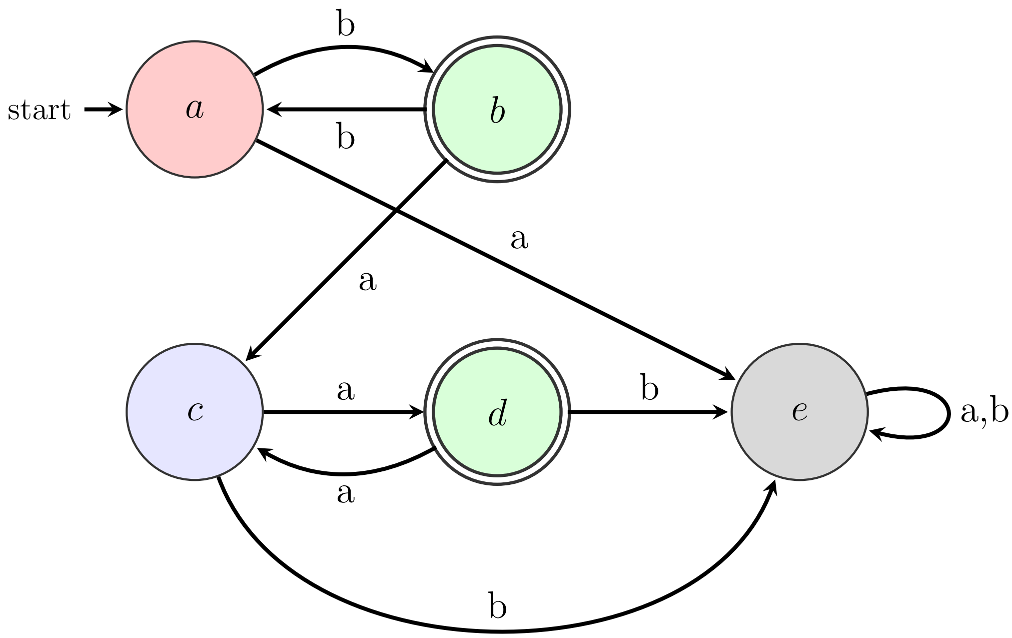
31.2 Even number of 1 and even number of 0



31.3 Odd number of 1



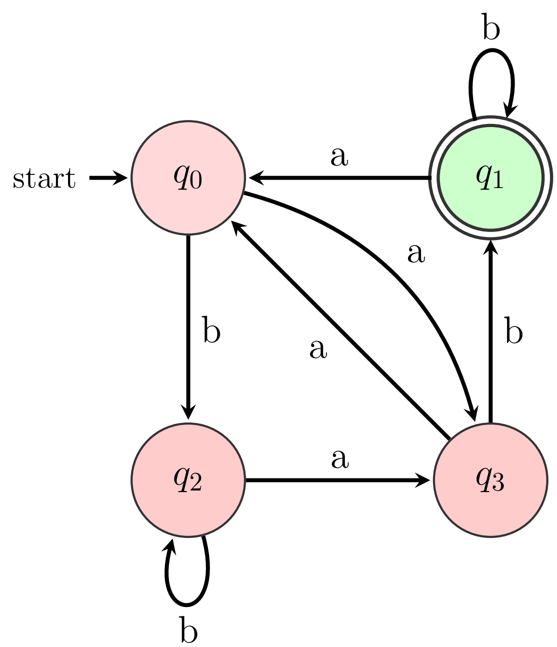
32 Even number of a and odd number of b and not containing substring ab



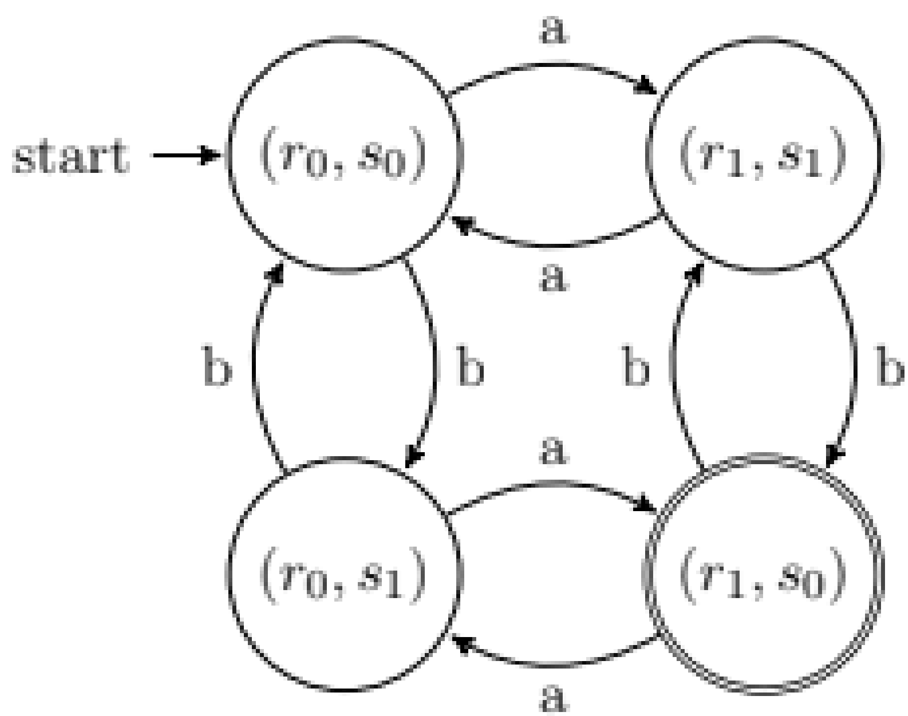
$$\mathbf{b(bb)^*(aa)^*}$$

33

$$L = \{\, w \mid w \textbf{ has an odd number of } a\textbf{'s and ends with a } b \,\}$$

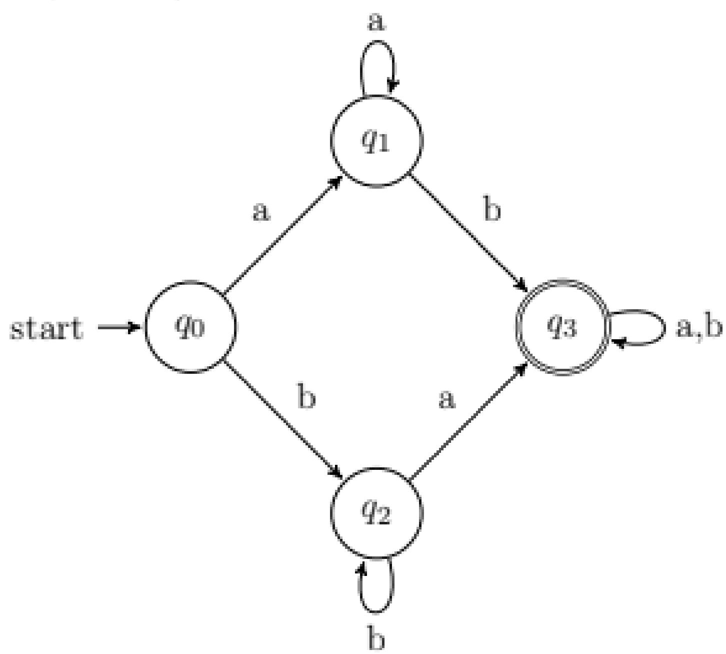

start
q_0
q_1
q_2
q_3
a
a
a
a
a
b
b
b
b

34 Has even length and an odd number of a's

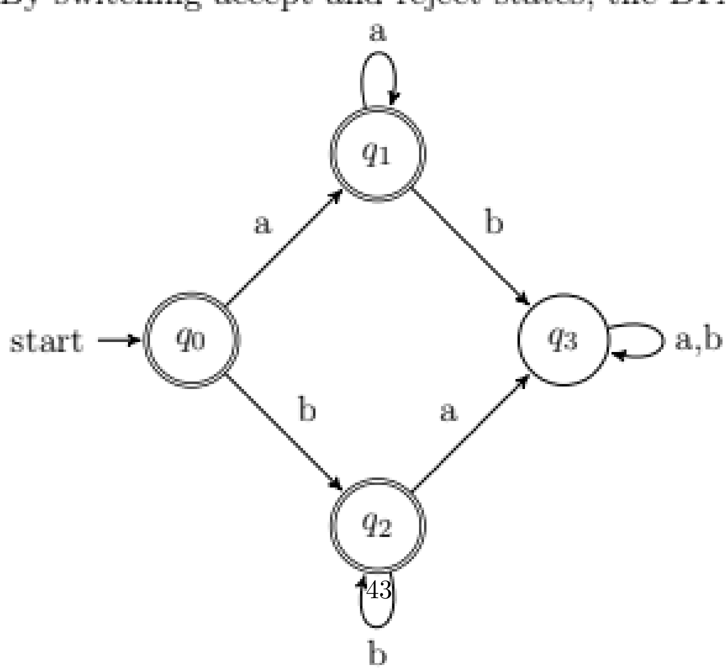


35 Qs

c) $L = \{w|w$ contains neither the substring `ab` nor `ba`$\}$
$\overline{L} = \{w|w$ contains either the substring `ab` or `ba`$\}$
The DFA for $\overline{L}$:

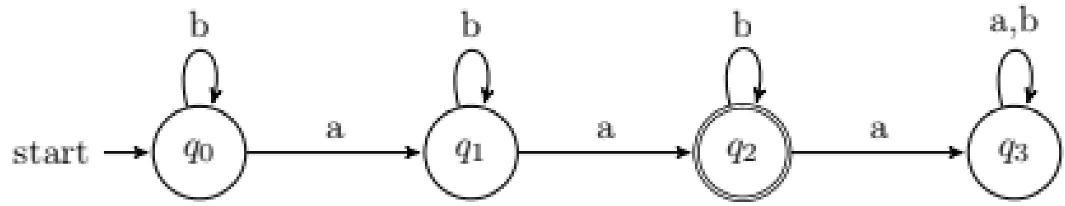


By switching accept and reject states, the DFA for L is as follows:



36 Qs

g) $L = \{w|w$ is any string that doesn't contain exactly two $\texttt{a}$s$\}$
$\overline{L} = \{w|w$ is any string that contains exactly two $\texttt{a}$s$\}$
The DFA for $\overline{L}$:



By switching accept and reject states, the DFA for L is as follows: