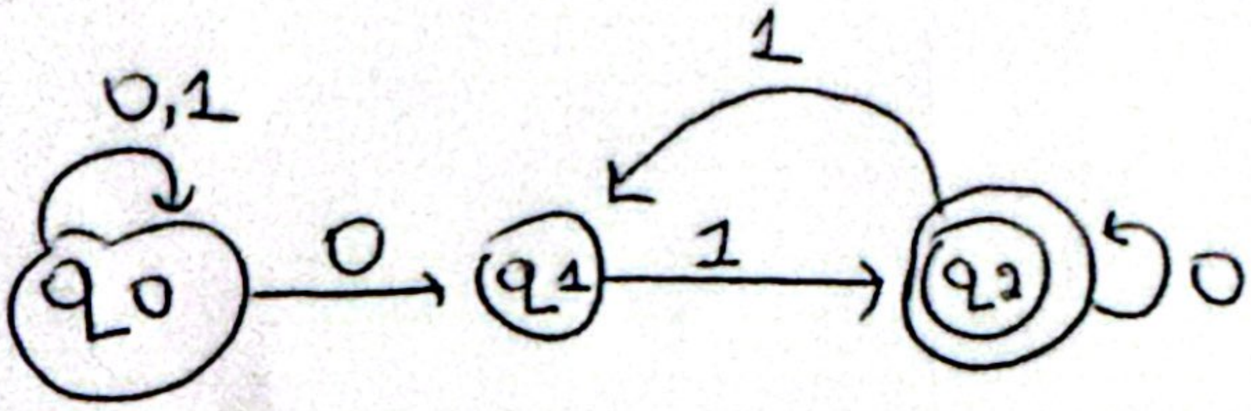
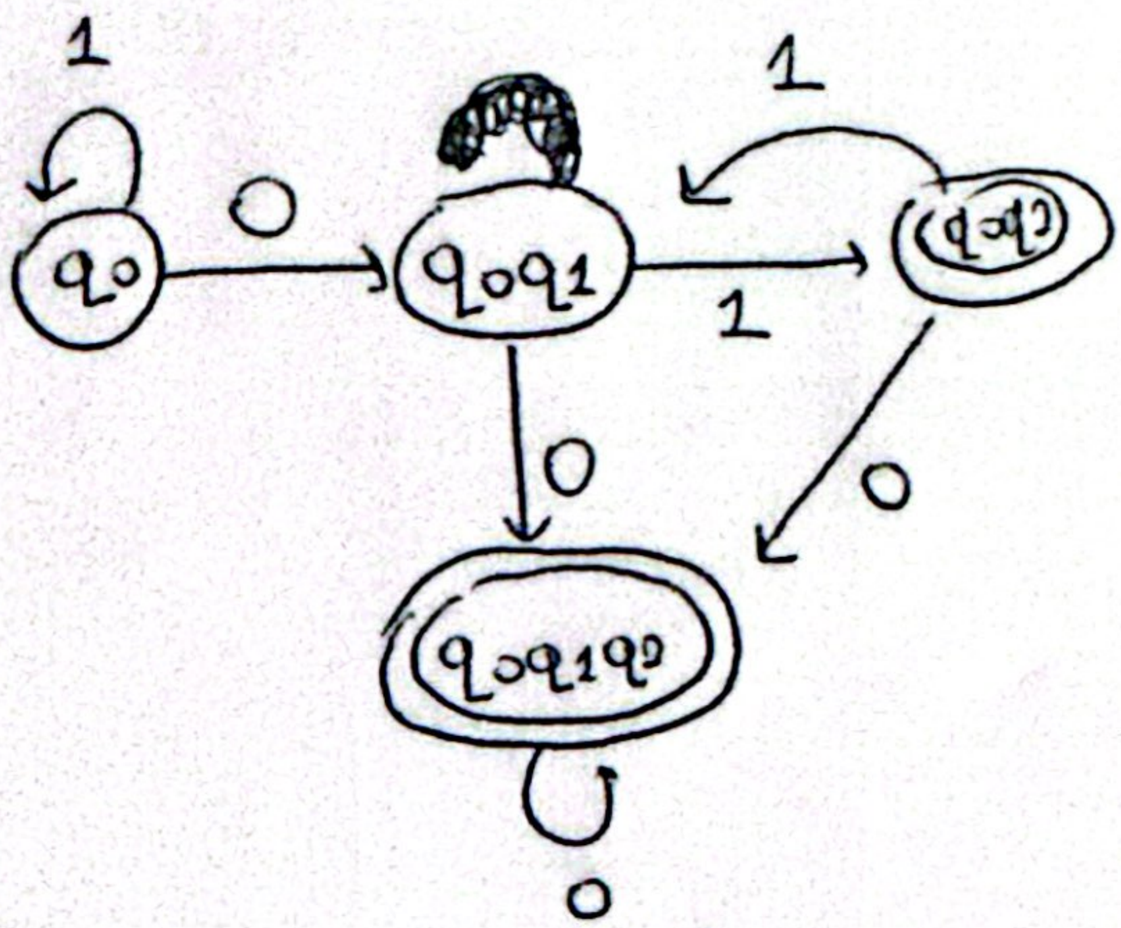
Q1	0	1
q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_2\}$	$\emptyset$
(q2)	$\{q_2\}$	$\{q_1\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$	$\{q_0, q_1, q_2\}$

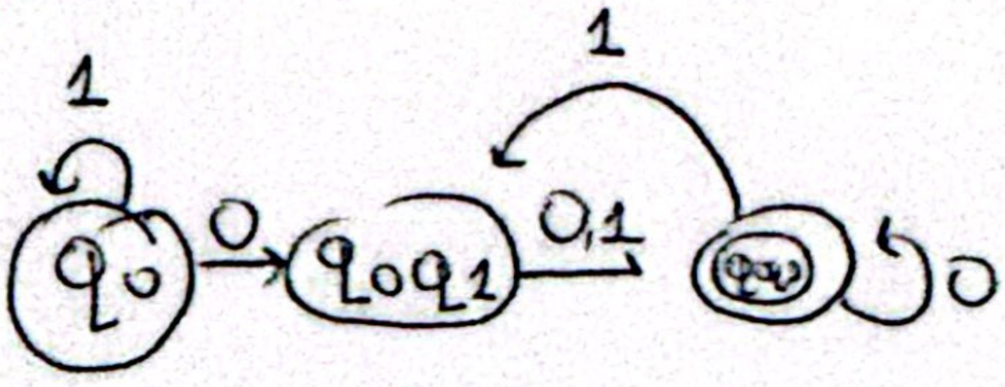
Initialy we are given the NFA



And we convert it to the DFA



And if we minimize it



Q2)

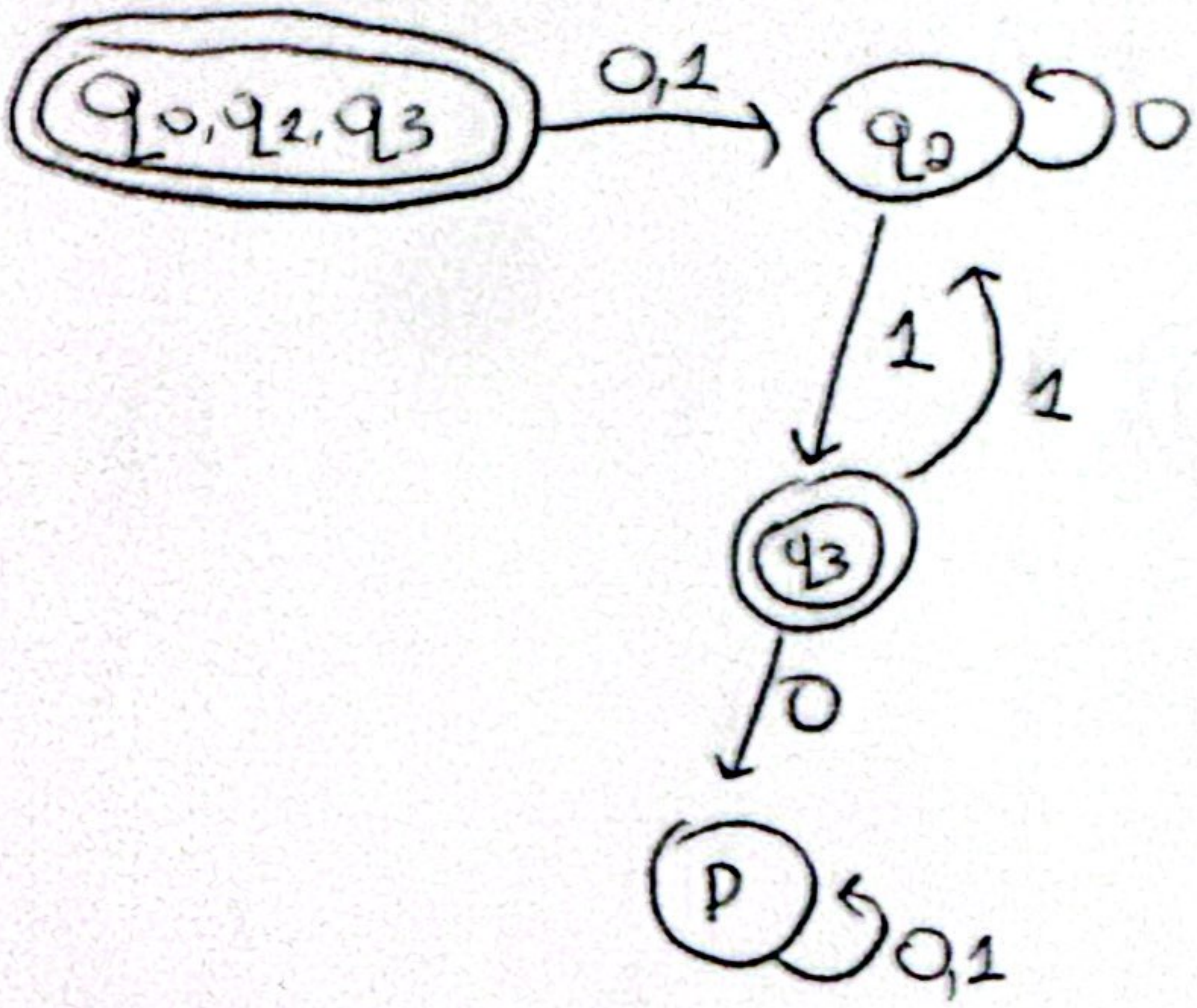
ϵ-closure(q_0) = {q_0, q_1, q_3}

ϵ-closure(q_2) = {q_2}

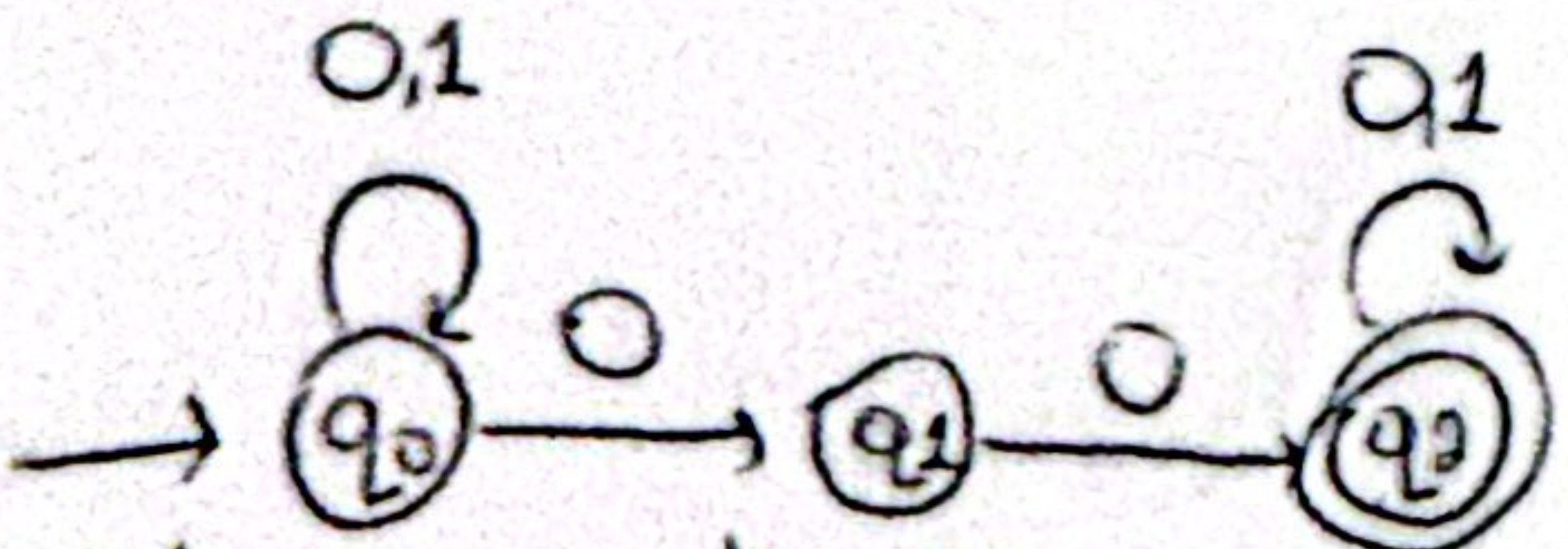
ϵ-closure([redacted] q_3) = {q_2 [redacted]}

~~ϵ-closure(q_3) = {q_3}~~

NFA	DFA	0	1
* {q_0, q_1, q_3}	A	B	[redacted] B
{q_2}	B	B	[redacted] C
[redacted]	[redacted]	[redacted]	[redacted]
* {q_3}	C [redacted]	D	[redacted] B
D = $\emptyset$		D	D

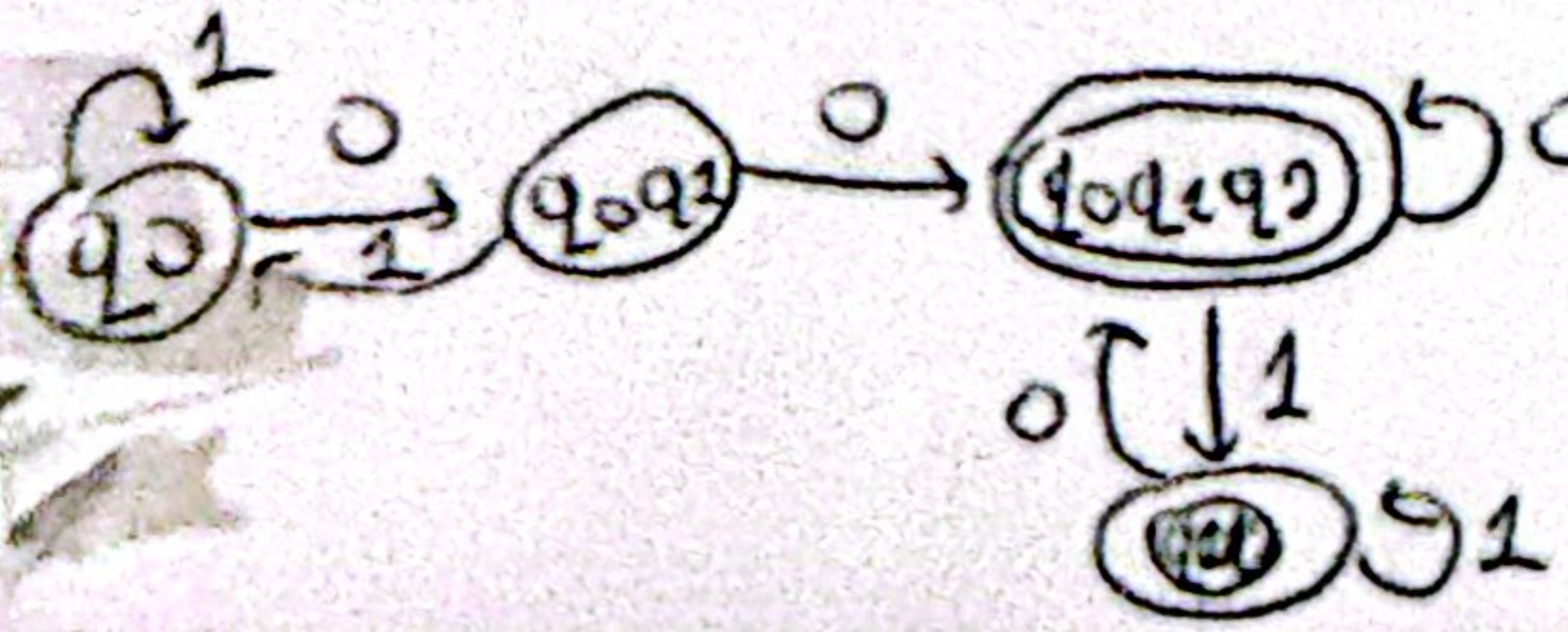


Q3) We require an NFA for the expression (0|1)* 00 (0|1)*

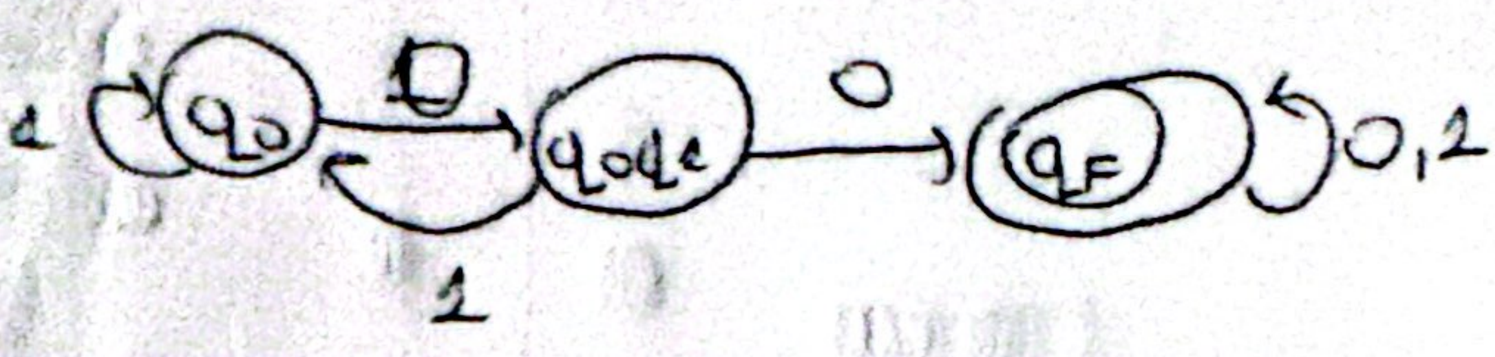


	0	1
q_0	{q_0, q_1}	{q_0}
(q_1	{q_2}	$\emptyset$)
*(q_2	{q_0}	{q_2})
q_0q_1	{q_0, q_1, q_2}	{q_0}
* q_0q_2	{q_0, q_1, q_2}	{q_0, q_2}
* {q_0, q_1, q_2}	{q_0, q_1, q_2}	{q_0, q_2}

And the corresponding DFA



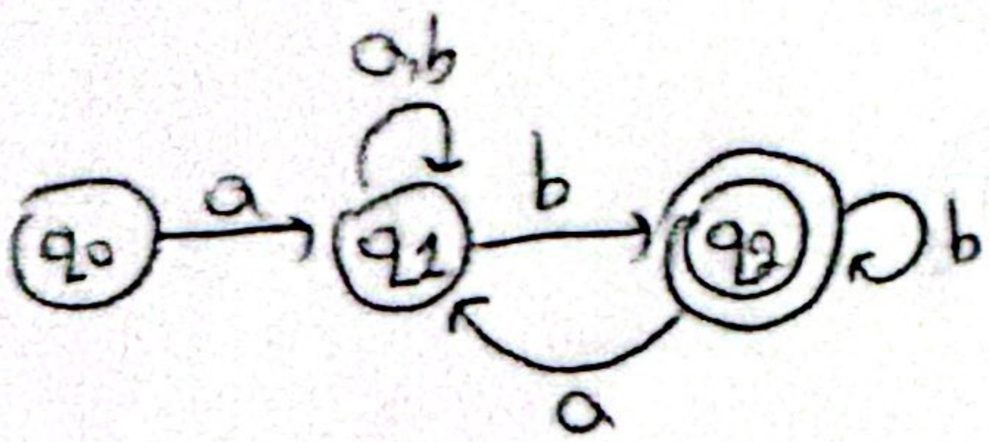
The two final states can be merged as follows



Q4) We require an RE for the language of strings over {a,b} that start with a and end with b

This can be achieved with a(a|b)*b

And the corresponding NFA is

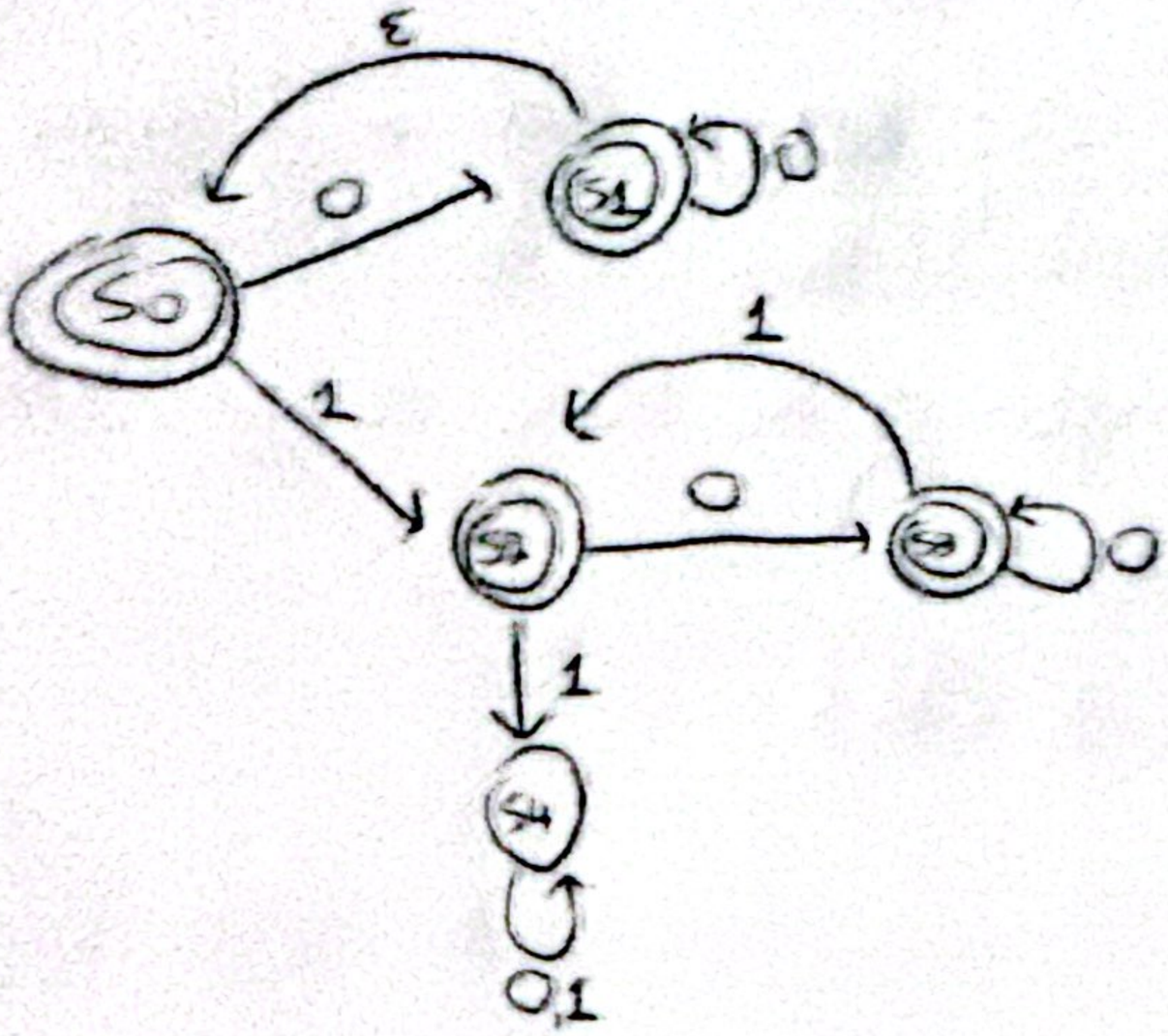


Q5)

i) We want to find the RE that accepts ore binary strings that do not contain consecutive 1s.

RE $(0|10)^*(1)?$

The NFA-ε that describes the RE is



$\in(S_0) = S_0$

$\in(S_1) = \{S_0, S_1\}$

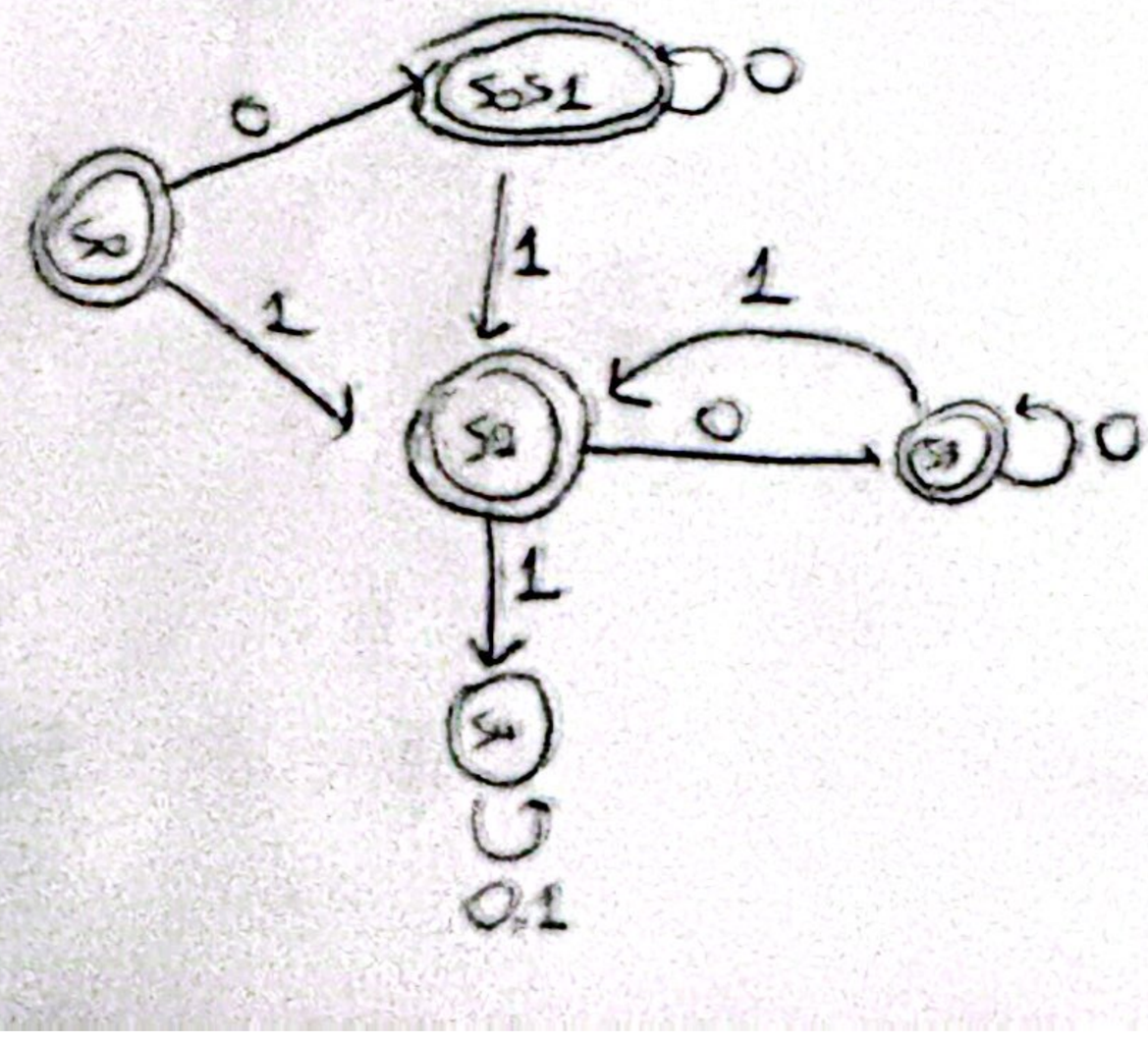
$\in(S_2) = \{S_2\}$

$\in(S_3) = \{S_3\}$

$\in(S_4) = \{S_4\}$

	0	1
$\{S_0 S_1\}$	$\{S_0 S_1\}$	$\{S_2\}$

The resulting DFA is



Q61

We are given the regular expression:

^([a-zA-z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$

The RE accepts strings of the format:

1) One of the following : a letter between a-z or A-z
   a number between 0-9
   the symbol _ (underscore)
   the symbol — (dash)
   the symbol . (dot)

The + means that it will select one of the above (not the same one each time) from 1 to infinite times.

2) Then follows the symbol @.

3) Then again follos the format from part 1)

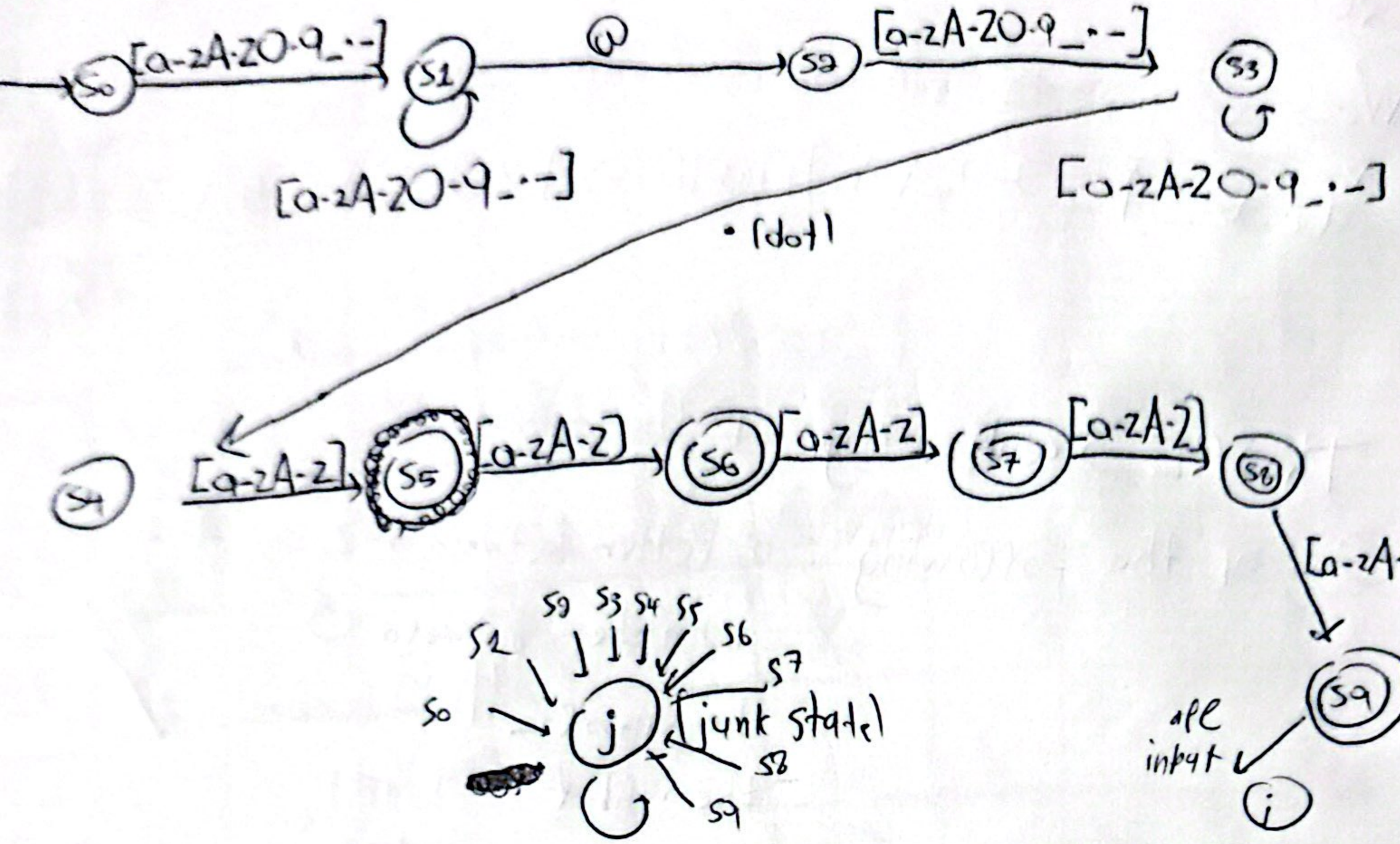
*4) Then the .(dot) character

5) Finally any letter (caps or not) from 2 to 5 times (it can be any letter each time)

^ and $ are the start and end of the string.

-

We want to design a DFA capable of accepting the above RE. The DFA we need is as follows:-



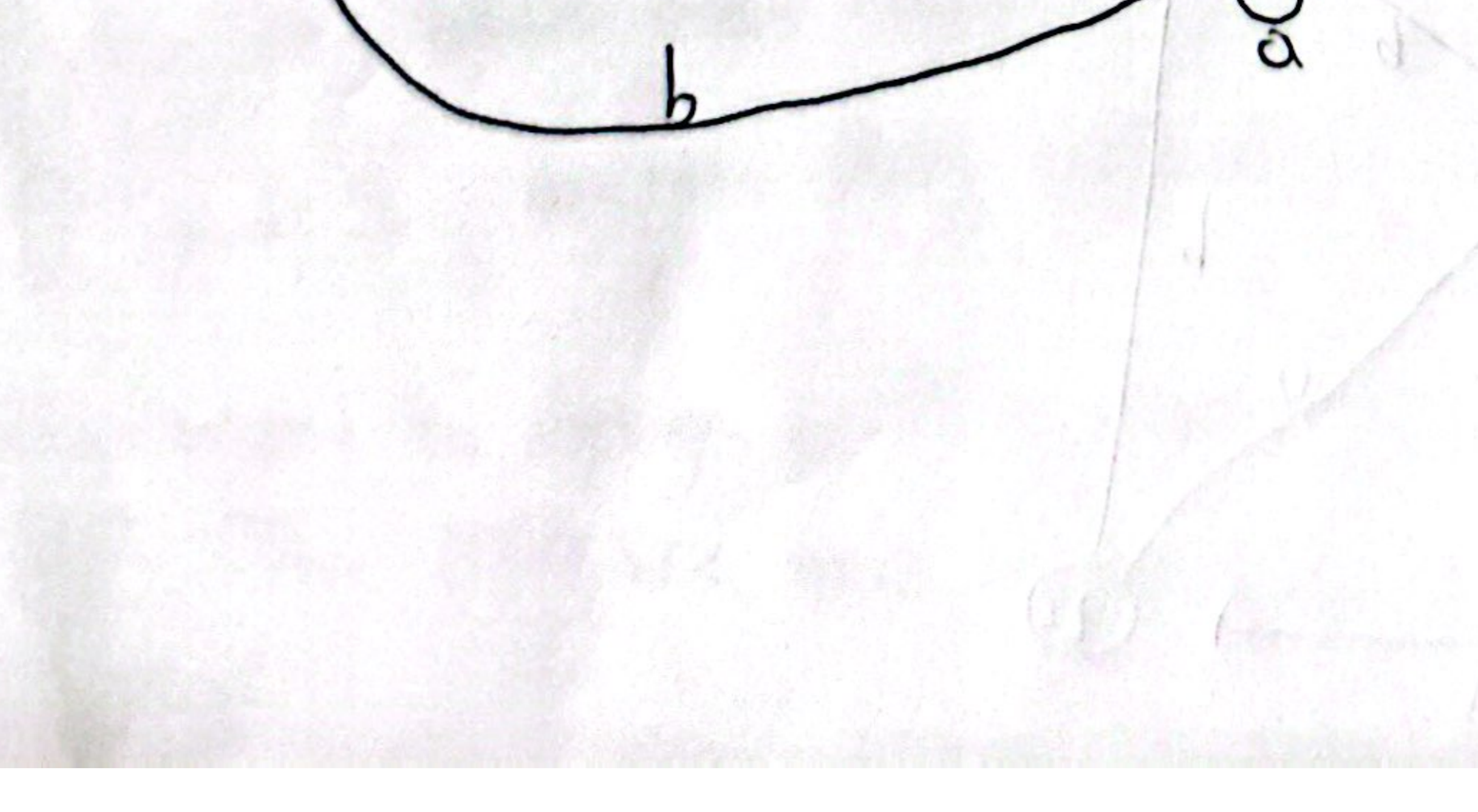
Every state goes to the trap state for any other input not specified in the transitions of the designed DFA. Once we are in the trap state all inputs lead to the same state.

Q7(i)

B	X_1						
C	X_0	X_0					
D		X_1	X_0				
E	X_1	X_1	X_0	X_1			
F	X_2	X_1	X_0	X_2	X_1		
G	X_1		X_0	X_1	X_1	X_1	
	A	B	C	D	E	F	

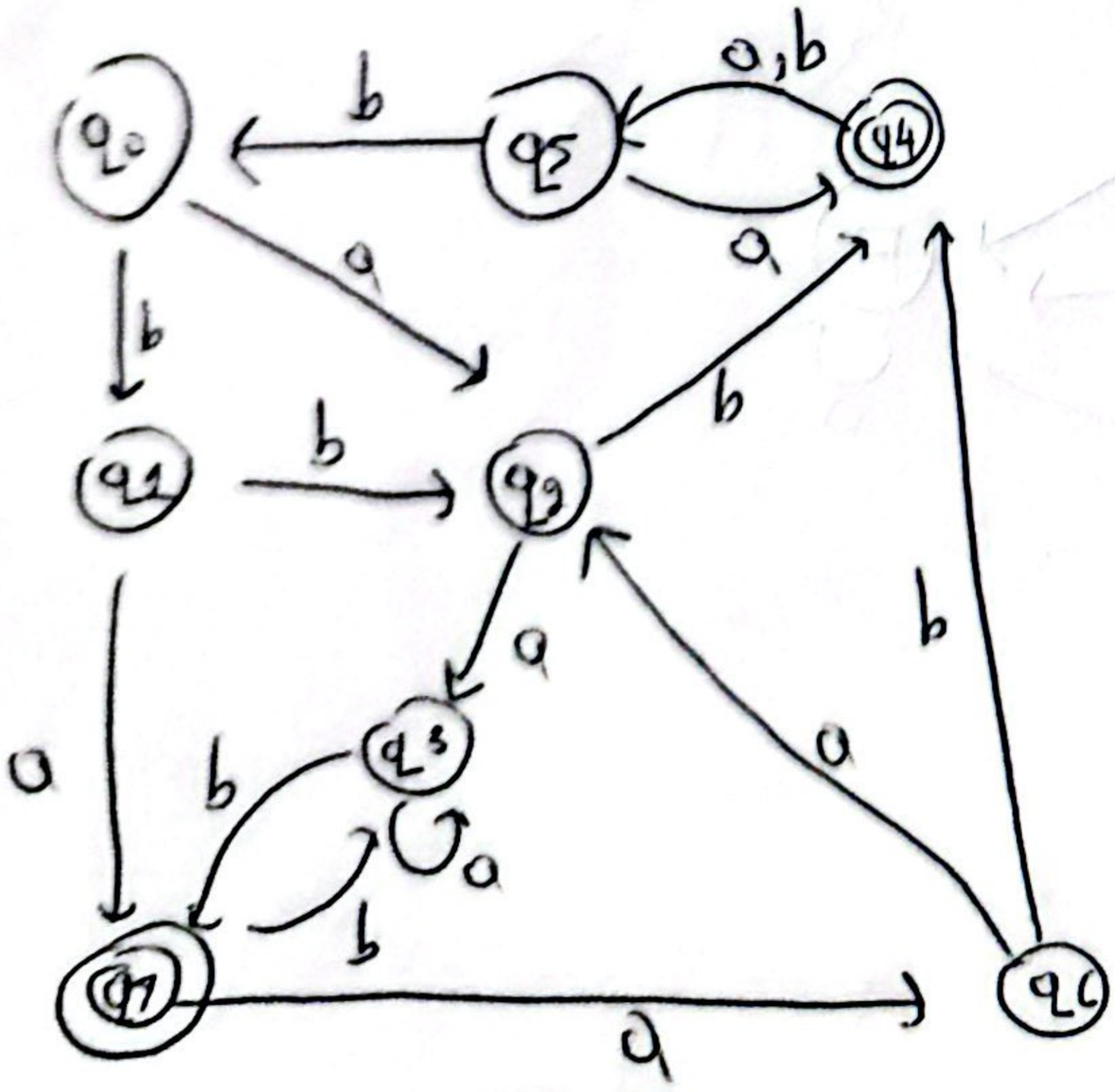
States

The minimized DFA is as follows:



q1	X_1							
q2	X_1	X_1						
q3	X_1	X_1	X_3					
(q4)	X_0	X_0	X_0	X_0				
q5	X_1	X_2	X_1	X_1	X_0			
q6	X_1	X_1	X_4	X_4	X_0	X_1		
(q7)	X_0	X_0	X_0	X_0	X_2	X_0	X_0	
q8	X_1	X_5	X_1	X_1	X_0	○	X_1	X_0
	q0	q1	q2	q3	(q4)	q5	q6	(q7)

- q5 and q8 are the only mergeable states. The final DFA is as follows



Q8) The RE describing a valid variable name is:

^([a-zA-Z_])([a-zA-Z0-9_]{0,30})$

We can analyze this RE as follows:

The first character of the string can be any upper or lower case letter or _ (underscore) ([a-zA-Z_])

Then we can have between 0-30 upper or lowercase letters or any single digit number* (letters and numbers can be different) so that the length of the name remains valid from 1-31 characters. (* or -)

The RE starts with ^ and ends with $.

- The corresponding DFA is: