# $\varepsilon$-NFA

- Epsilon transitions are transitions that occur without consuming any input symbols.

- In other words, they allow the NFA to transition from one state to another without reading any input.

- This can be useful in situations where multiple paths could lead to the same result.

- By using epsilon transitions, the NFA can effectively explore all possible paths before making a decision.

- A NFA can be represented by a 5-tuple $(Q, \sum, \delta, q_0, F)$ where –

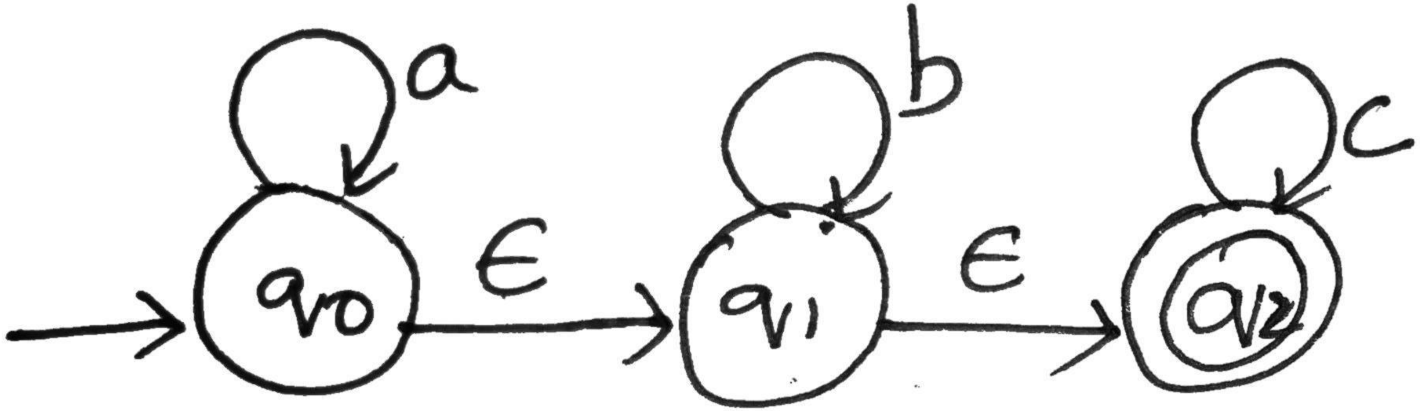
  **Q** is a finite set of states.

  $\sum$ is a finite set of symbols called the alphabet.

  **δ** is the transition function where $\delta: Q \times( \sum \cup \{\in\}) \rightarrow 2^Q$

  $\mathbf{q_0}$ is the initial state from where any input is $(q_0 \in Q)$.

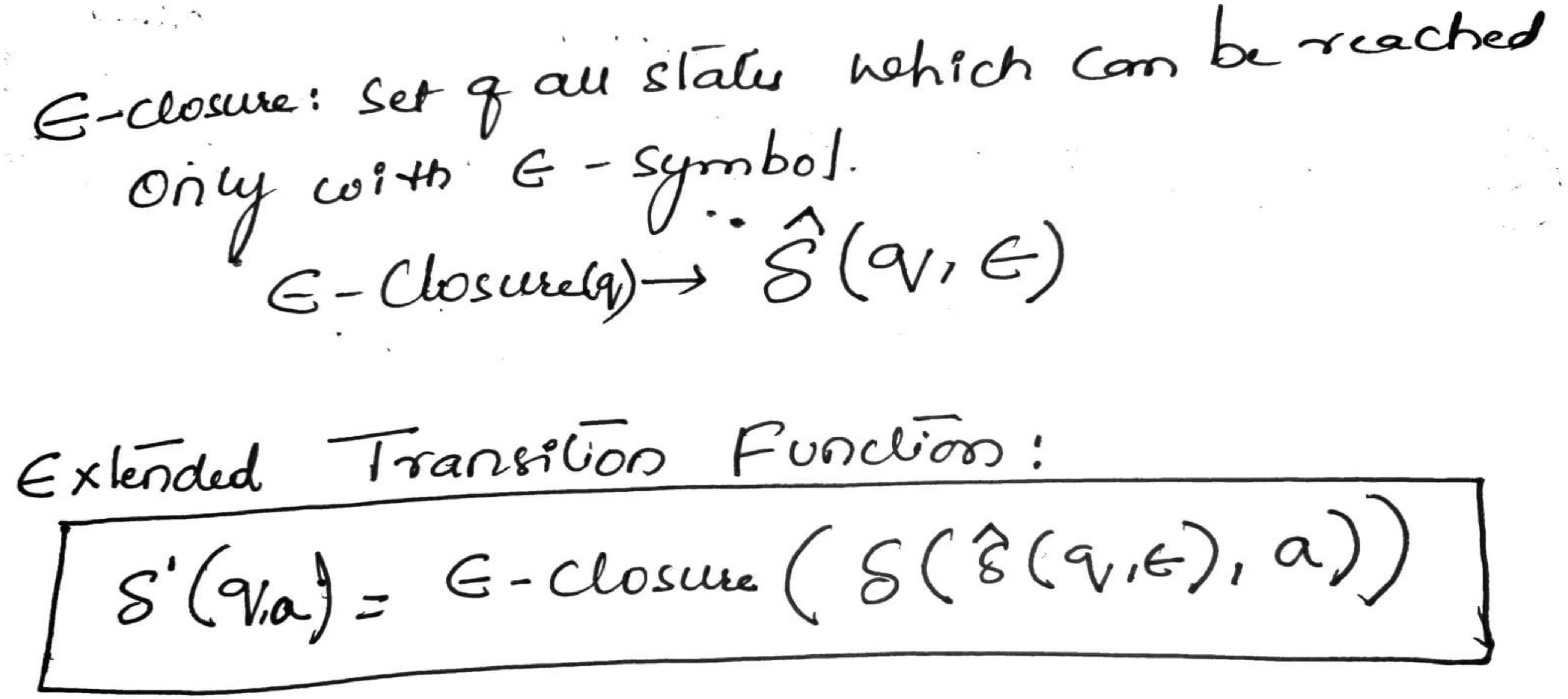
  **F** is a set of final state/states of Q $(F \subseteq Q)$.



Only 'a' accepted since $a \epsilon \epsilon$ $q_2$

Only 'b' accepted since $b \epsilon$ $q_2$

# $\varepsilon$-closure



$\in$-closure: Set of all states which can be reached only with $\in$-symbol.

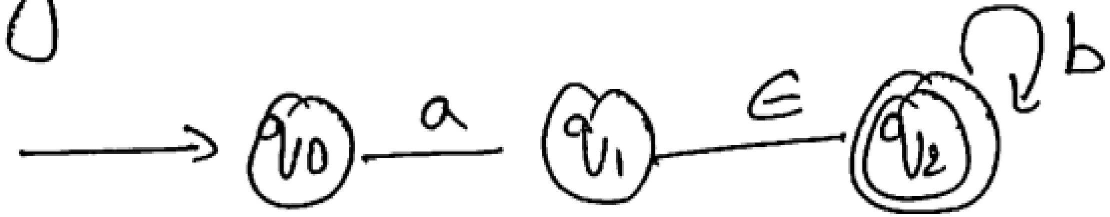
$\therefore$ $\in$-Closure(q) $\rightarrow$ $\hat{\delta}(q, \in)$

Extended Transition Function:

$$\boxed{\delta'(q,a) = \in\text{-closure}(\delta(\hat{\delta}(q,\in), a))}$$

# ε-closure Examples:

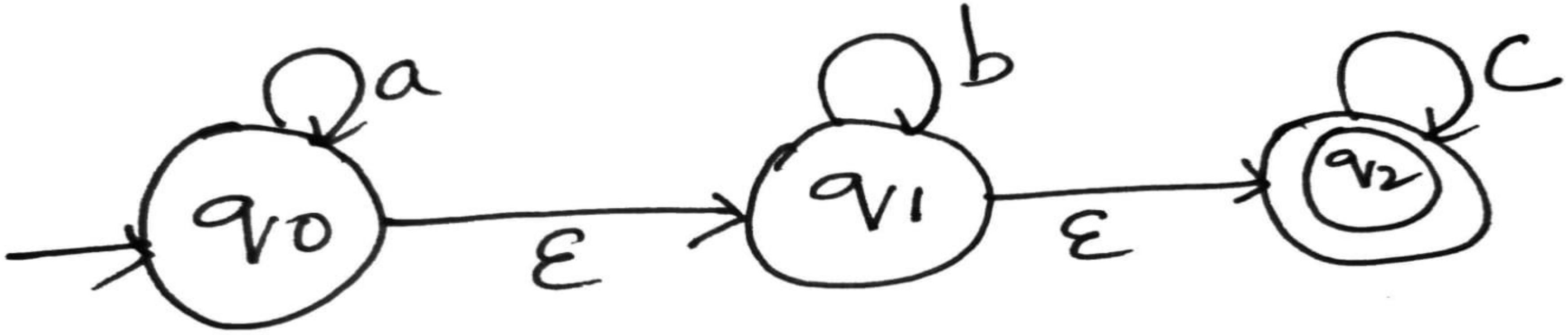
① → Find $\in$-closures of all states for the following diagram.



$\in$-closure$(q_0)$ = $\{q_0\}$

$\in$-closure$(q_1)$ = $\{q_1, q_2\}$

$\in$-closure$(q_2)$ = $\{q_2\}$

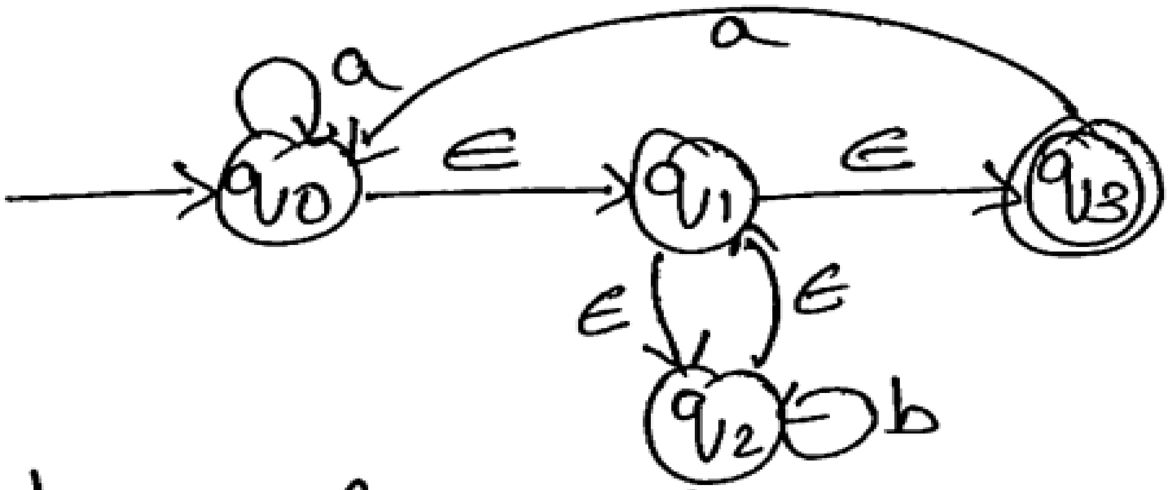


$\epsilon$ - Closure$(q_0) = \{q_0, q_1 q_2\}$

$\epsilon$ - Closure$(q_1) = \{q_1 q_2\}$

$\epsilon$ - Closure $\{q_2) = \{q_2\}$

(2) →



$\in$- closure $(q_0) = \{q_0, q_1, q_2, q_3\}$

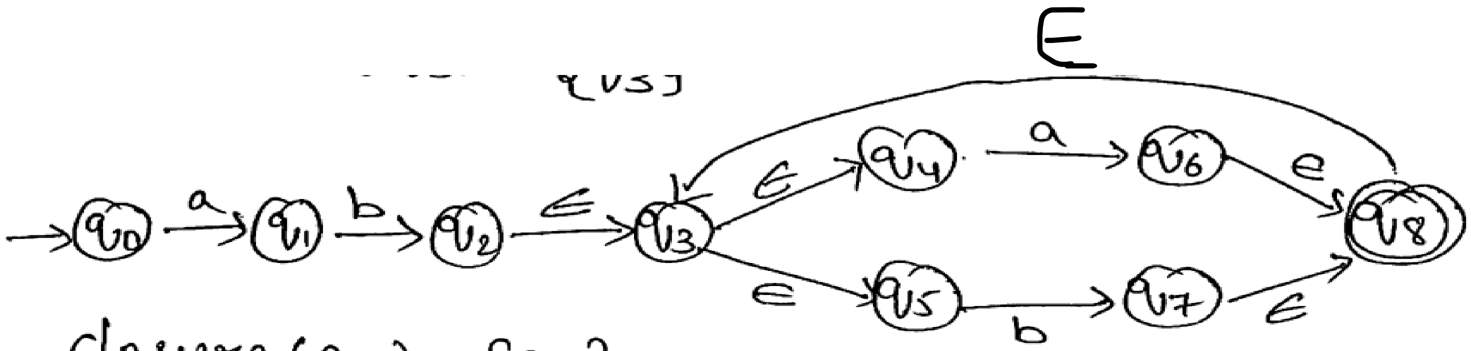
$\in$-closure $(q_1) = \{q_1, q_2, q_3\}$

$\in$-closure $(q_2) = \{q_1, q_2, q_3\}$

$\in$- closure $(q_3) = \{q_3\}$

$\{q_5\}$

③



$\epsilon$ - closure $(q_0) = \{q_0\}$

$\epsilon$ - closure $(q_1) = \{q_1\}$

$\epsilon$ - closure $(q_2) = \{q_2, q_3, q_4, q_5\}$

$\epsilon$ - closure $(q_3) = \{q_3, q_4, q_5\}$

$\epsilon$ - closure $(q_4) = \{q_4\}$

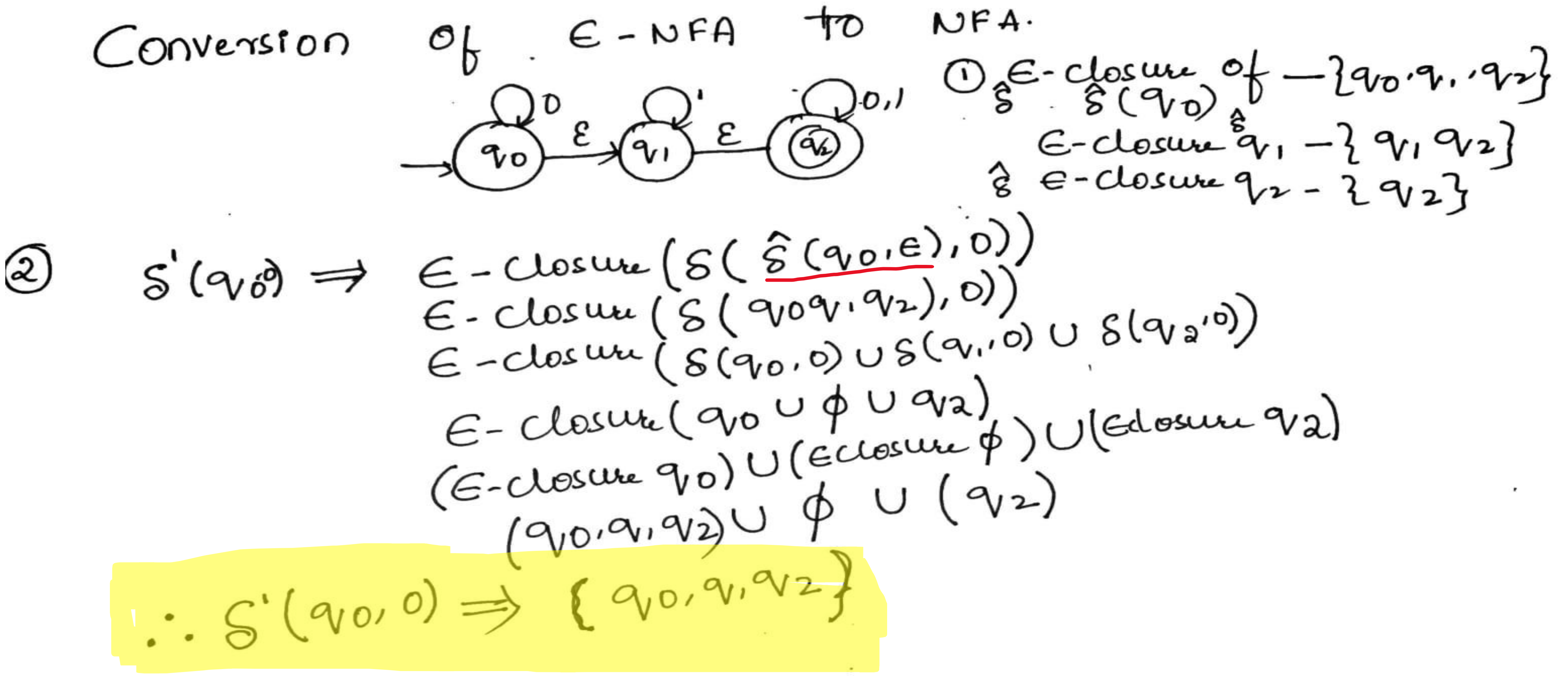
$\epsilon$ - closure $(q_5) = \{q_5\}$

$\epsilon$ - closure $(q_6) = \{q_6, q_8, q_3, q_4, q_5\}$

$\epsilon$ - closure $(q_7) = \{q_7, q_8, q_3, q_4, q_5\}$

$\epsilon$ - closure $(q_8) = \{q_8, q_3, q_4, q_5\}$

# Conversion $\varepsilon$-NFA to NFA


0
$\epsilon$
1
$\epsilon$
0,1
$q_0$
$q_1$
$q_2$

Conversion of $\epsilon$-NFA to NFA.

① $\hat{\delta}$ $\epsilon$-closure $\hat{\delta}(q_0)$ of $-\{q_0, q_1, q_2\}$

$\epsilon$-closure $\hat{\delta}$ $q_1 - \{q_1, q_2\}$

$\hat{\delta}$ $\epsilon$-closure $q_2 - \{q_2\}$

② $\delta'(q_0, 0) \Rightarrow \epsilon\text{-closure}(\delta(\underline{\hat{\delta}(q_0, \epsilon)}, 0))$

$\epsilon\text{-closure}(\delta(q_0 q_1, q_2), 0))$

$\epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$\epsilon\text{-closure}(q_0 \cup \phi \cup q_2)$

$(\epsilon\text{-closure } q_0) \cup (\epsilon\text{closure } \phi) \cup (\epsilon\text{closure } q_2)$

$(q_0, q_1, q_2) \cup \phi \cup (q_2)$

$\therefore \delta'(q_0, 0) \Rightarrow \{q_0, q_1, q_2\}$

$$\delta'(q_0,1) \Rightarrow \epsilon\text{-closure}\left(\delta\left(\underline{\hat{\delta}(q_0,\epsilon)},1\right)\right)$$

$$\epsilon\text{-closure}\left(\delta\left(q_0 q_1 q_2\right),1\right))$$

$$\epsilon\text{-closure}\left(\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)\right)$$

$$\epsilon\text{-closure}\left(\phi \cup q_1 \cup q_2\right)$$

$$\epsilon\text{-closure}\left(\phi\right) \cup \epsilon\text{-closure}(q_1) \cup \epsilon\text{-closure}(q_2)$$

$$\Rightarrow \phi \cup \{q_1 q_2\} \cup \{q_2\}$$

$$\therefore \delta'(q_0,1) \Rightarrow \{q_1 q_2\}$$

$$\delta'(q_1, 0) \Rightarrow \epsilon\text{-closure}(\delta(\underline{\hat{\delta}(q_1, \epsilon)}, 0))$$

$$\Rightarrow \epsilon\text{-closure}(\delta(q_1 q_2), 0))$$

$$\Rightarrow \epsilon\text{-Closure}(\delta(q_1, 0) \cup \delta(q_2, 0))$$

$$\Rightarrow \epsilon\text{-closure}(\phi \cup q_2)$$

$$\Rightarrow \epsilon\text{-closure}(\phi) \cup \epsilon\text{-closure}(q_2)$$

$$\Rightarrow \phi \cup q_2$$

$$\therefore \delta'(q_1, 0) \Rightarrow \{q_2\}$$

$$\delta'(q_1, 1) \Rightarrow \epsilon\text{-closure}(\delta(\underline{\hat{\delta}(q_1, \epsilon)}, 1))$$

$$\Rightarrow \epsilon\text{-closure}(\delta(q_1 q_2), 1))$$

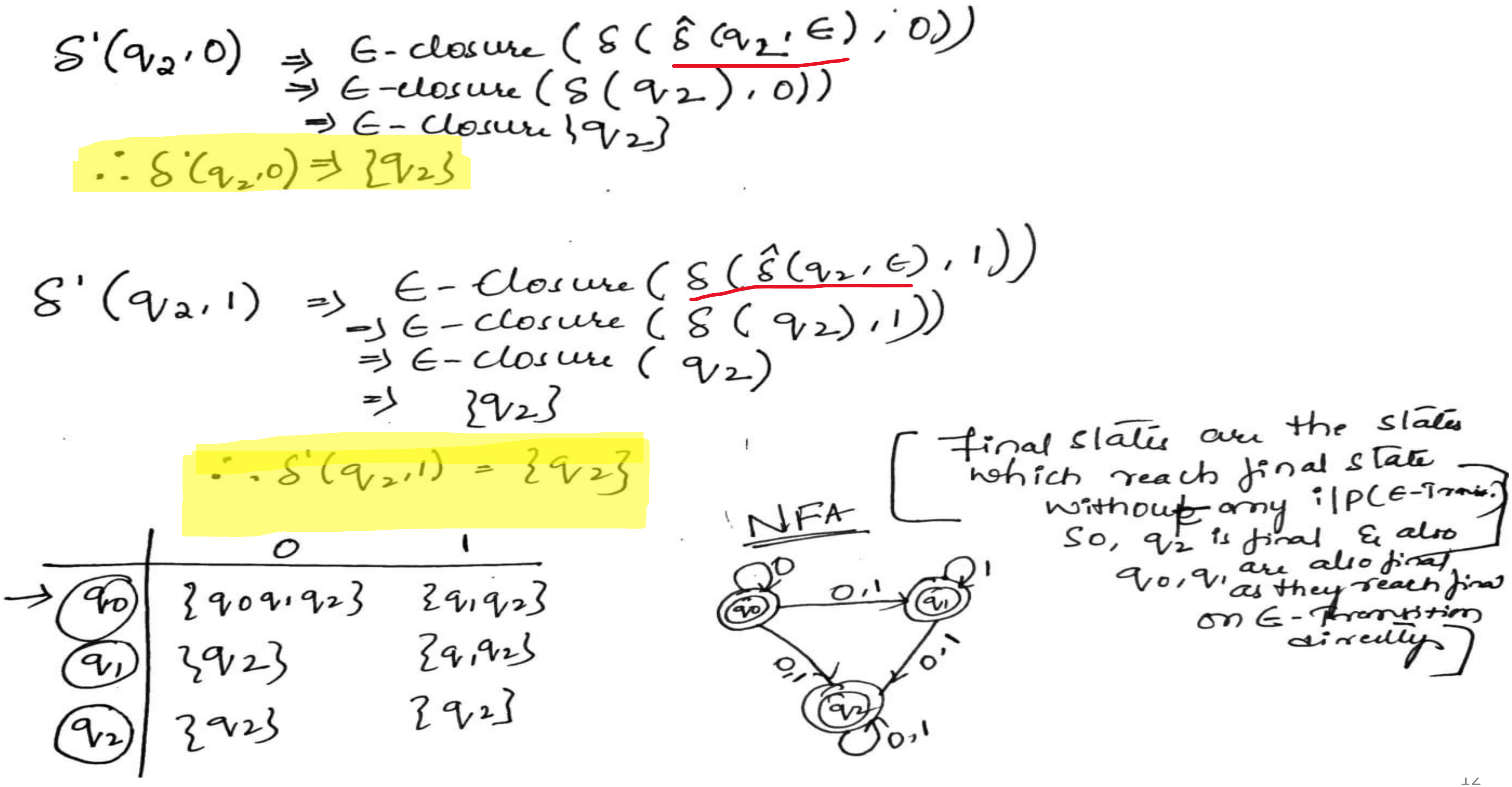
$$\Rightarrow \epsilon\text{-closure}(\delta(q_1, 1) \cup \delta(q_2, 1))$$

$$\Rightarrow \epsilon\text{-closure}(q_1 \cup q_2)$$

$$\Rightarrow \epsilon\text{-closure}(q_1) \cup \epsilon\text{-closure}(q_2)$$

$$\Rightarrow \{q_1 q_2\} \cup \{q_2\}$$

$$\therefore \delta'(q_1, 1) = \{q_1, q_2\}$$



$\delta'(q_2,0) \Rightarrow \epsilon\text{-closure}(\delta(\underline{\hat{\delta}(q_2,\epsilon)},0))$

$\Rightarrow \epsilon\text{-closure}(\delta(q_2),0))$

$\Rightarrow \epsilon\text{-closure}\{q_2\}$

$\therefore \delta'(q_2,0) \Rightarrow \{q_2\}$

$\delta'(q_2,1) \Rightarrow \epsilon\text{-Closure}(\underline{\delta(\hat{\delta}(q_2,\epsilon)},1))$

$\Rightarrow \epsilon\text{-closure}(\delta(q_2),1))$

$\Rightarrow \epsilon\text{-closure}(q_2)$

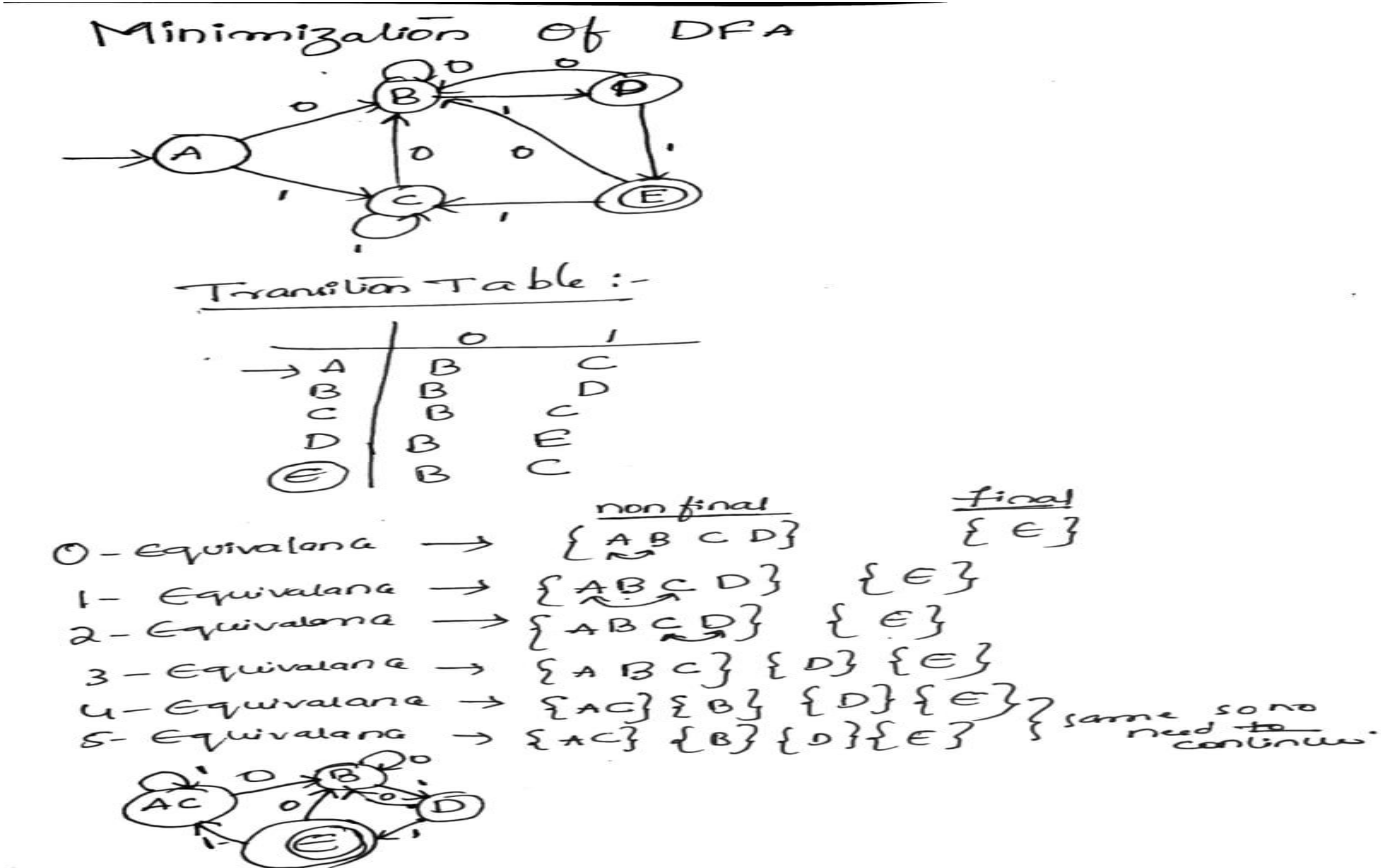
$\Rightarrow \{q_2\}$

$\therefore \delta'(q_2,1) = \{q_2\}$

| | 0 | 1 |
|---|---|---|
| → $q_0$ | $\{q_0 q_1, q_2\}$ | $\{q_1, q_2\}$ |
| $q_1$ | $\{q_2\}$ | $\{q_1, q_2\}$ |
| $q_2$ | $\{q_2\}$ | $\{q_2\}$ |

final states are the states which reach final state without any i/p(ε-Transi.)
So, $q_2$ is final & also $q_0, q_1$ are also final, as they reach final on ε-transition directly

# Minimization of DFA



Minimization of DFA

Transition Table :-

| | 0 | 1 |
|---|---|---|
| → A | B | C |
| B | B | D |
| C | B | C |
| D | B | E |
| E | B | C |

non final final

0 - Equivalance → {A B C D} {E}

1 - Equivalance → {A B C D} {E}

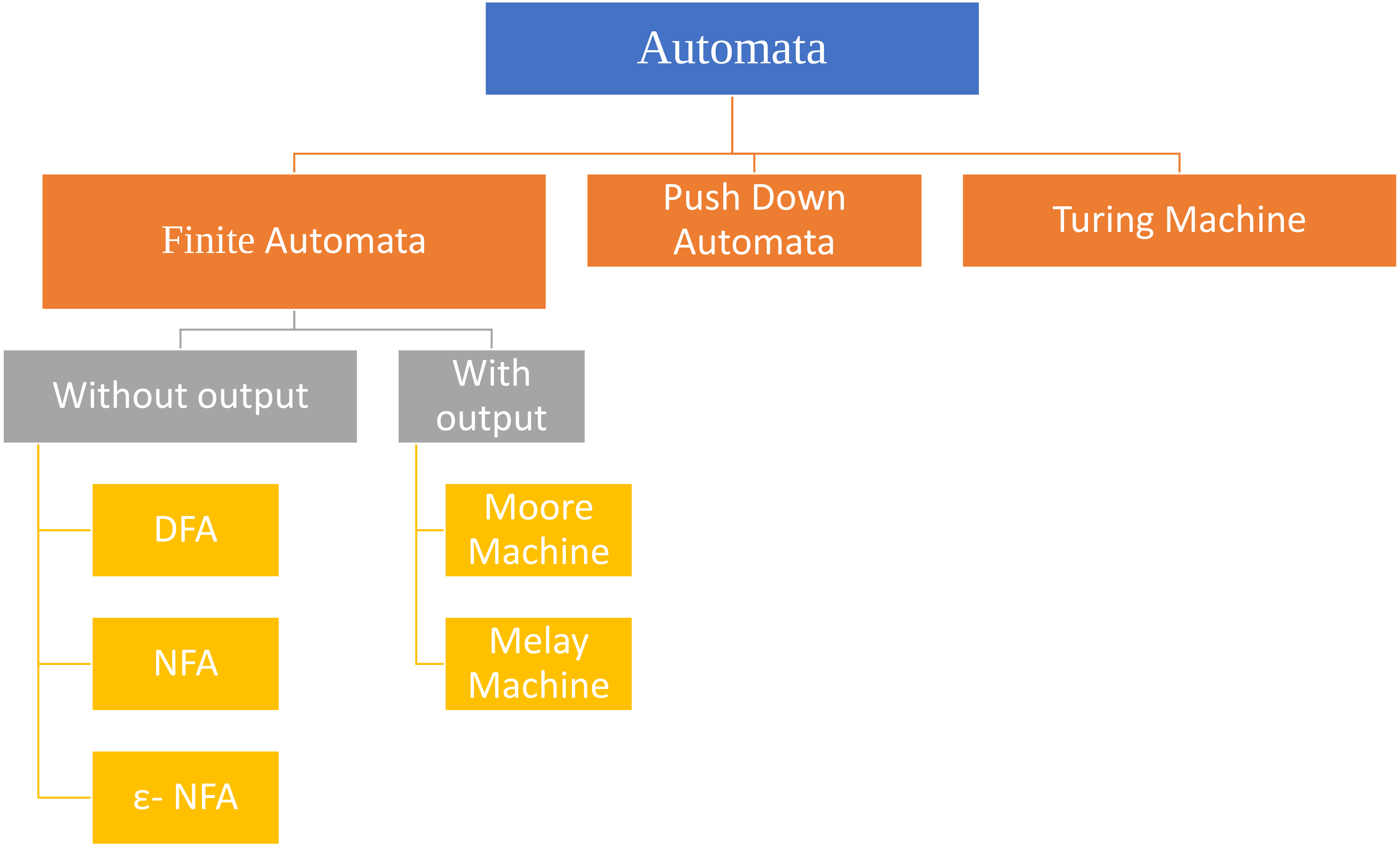
2 - Equivalance → {A B C D} {E}

3 - Equivalance → {A B C} {D} {E}

4 - Equivalance → {AC} {B} {D} {E}

5 - Equivalance → {AC} {B} {D} {E}

same so no need ~~to~~ continue.


Automata
Finite Automata
Push Down Automata
Turing Machine
Without output
With output
DFA
NFA
ε- NFA
Moore Machine
Melay Machine

# Moore Machine

- A **Moore machine** is a finite state machine that has an output value rather than a final state.

- For a given input, the machine generates a corresponding output.

- The output of the Moore machine depends only on the present state of the FA.

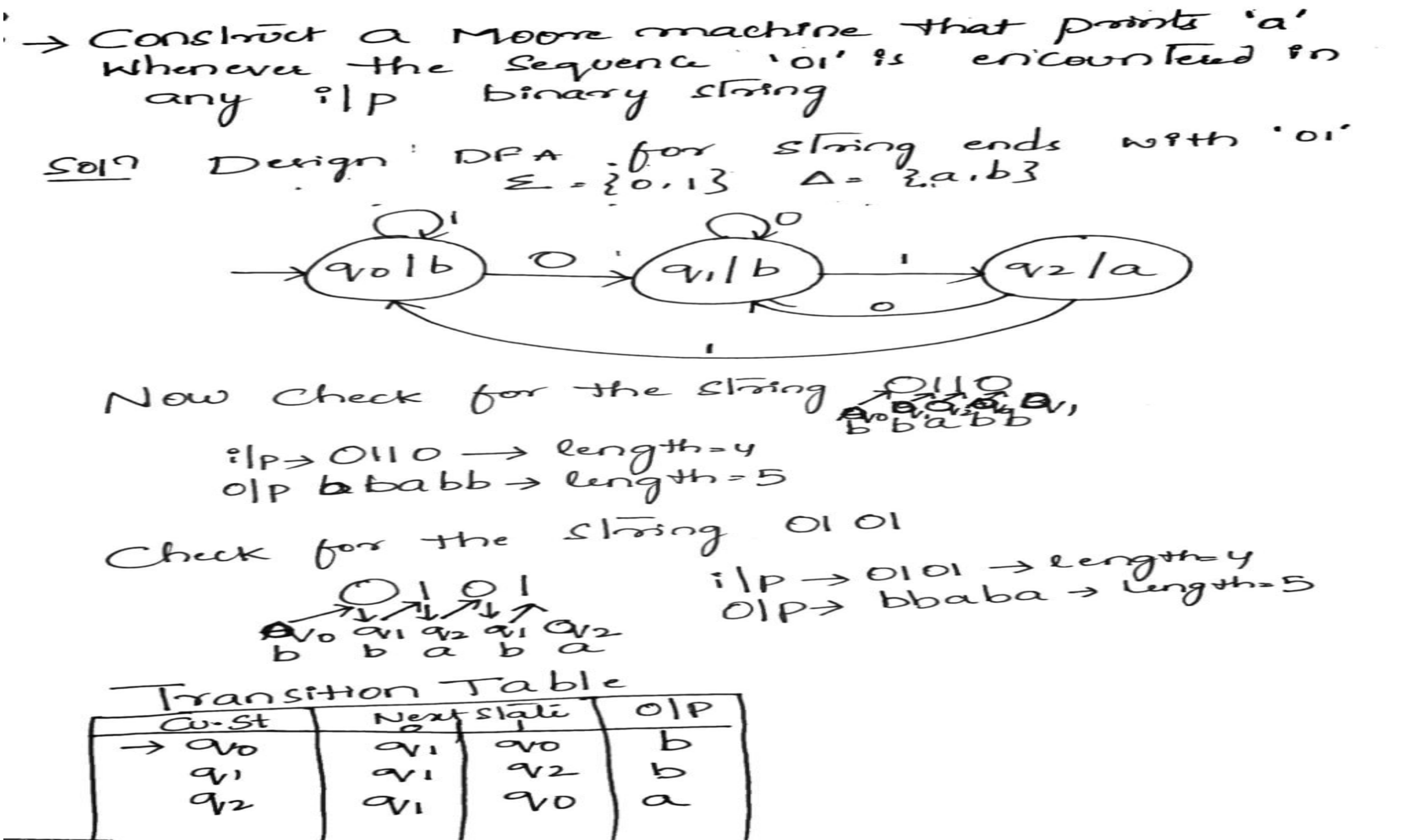
- Unlike other finite automata that determine the acceptance of a particular string in a given language, Moore machines determine the output against given input.

# Formal Definition Of Moore Machine

The Moore machine is a 6 tuple machine $(Q,\Sigma,q0,\Delta,\delta,\lambda)$:

$Q$: This is a set of states.
$\Sigma$: This is a set of input alphabets.
$q0$: This is the initial state.
$\Delta$: This is a set of output states.
$\delta$: This is a transition function, that is: $Q\times\Sigma\rightarrow Q$
$\lambda$: This is an output function, that is: $Q\rightarrow\Delta$

The output function means that for every state there is a corresponding output associated with it.


q0/b
q1/b
q2/a
1
0
0
1
0
1

→ Construct a Moore machine that prints 'a' whenever the sequence '01' is encountered in any i/p binary string

Soln Design DFA for string ends with '01'

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

Now Check for the string 0110

$q_0 \; q_1 \; q_2 \; q_0 \; q_1$

b b a b b

i/p → 0110 → length = 4

o/p bbabb → length = 5

Check for the string 0101

0 1 0 1

$q_0 \; q_1 \; q_2 \; q_1 \; q_2$

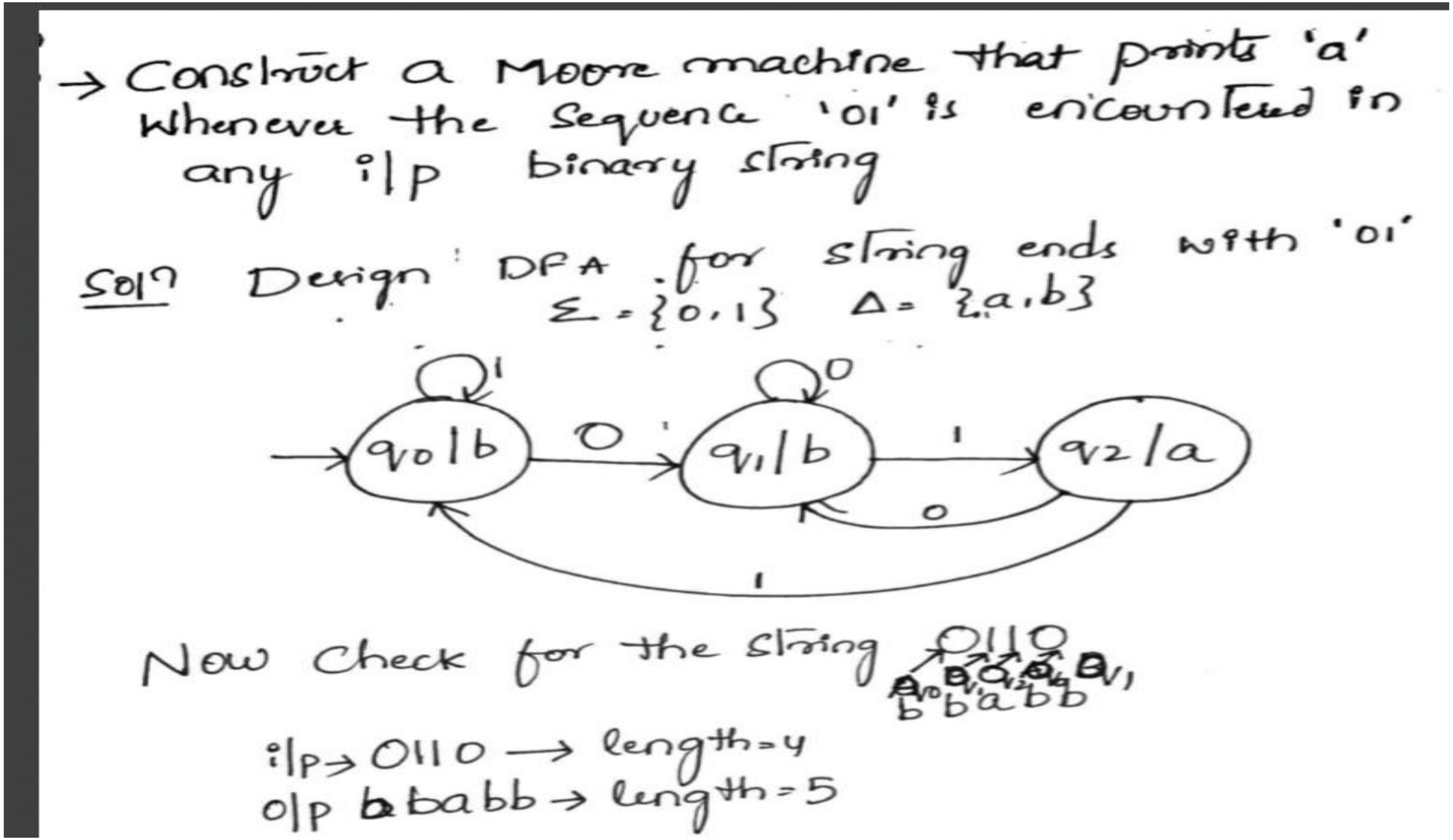
b b a b a

i/p → 0101 → length = 4

O/P → bbaba → length = 5

Transition Table

| Cu. St | Next State 0 | Next State 1 | O/P |
|---|---|---|---|
| → $q_0$ | $q_1$ | $q_0$ | b |
| $q_1$ | $q_1$ | $q_2$ | b |
| $q_2$ | $q_1$ | $q_0$ | a |



→ Construct a Moore machine that prints 'a' whenever the sequence '01' is encountered in any i|p binary string

Sol<sup></sup>n Design : DFA for string ends with '01'

$\Sigma = \{0,1\}$  $\Delta = \{a,b\}$

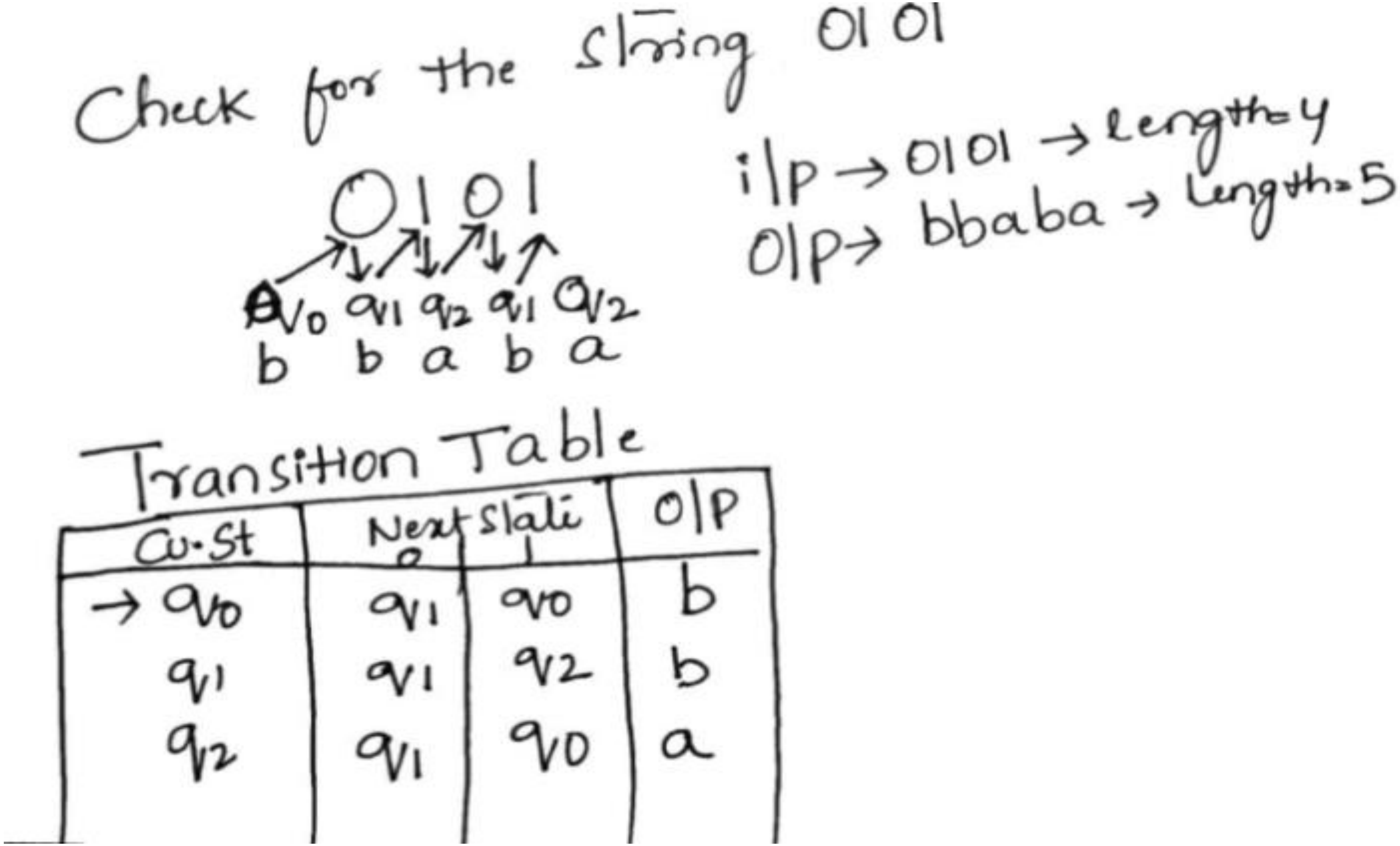
Now Check for the string 0110

$q_0\ q_1\ q_2\ q_0\ q_1$

b b a b b

i|p → 0110 → length = 4

o|p ~~b~~ babb → length = 5



Check for the string 0101

i/p → 0101 → length=4

o/p → bbaba → length=5

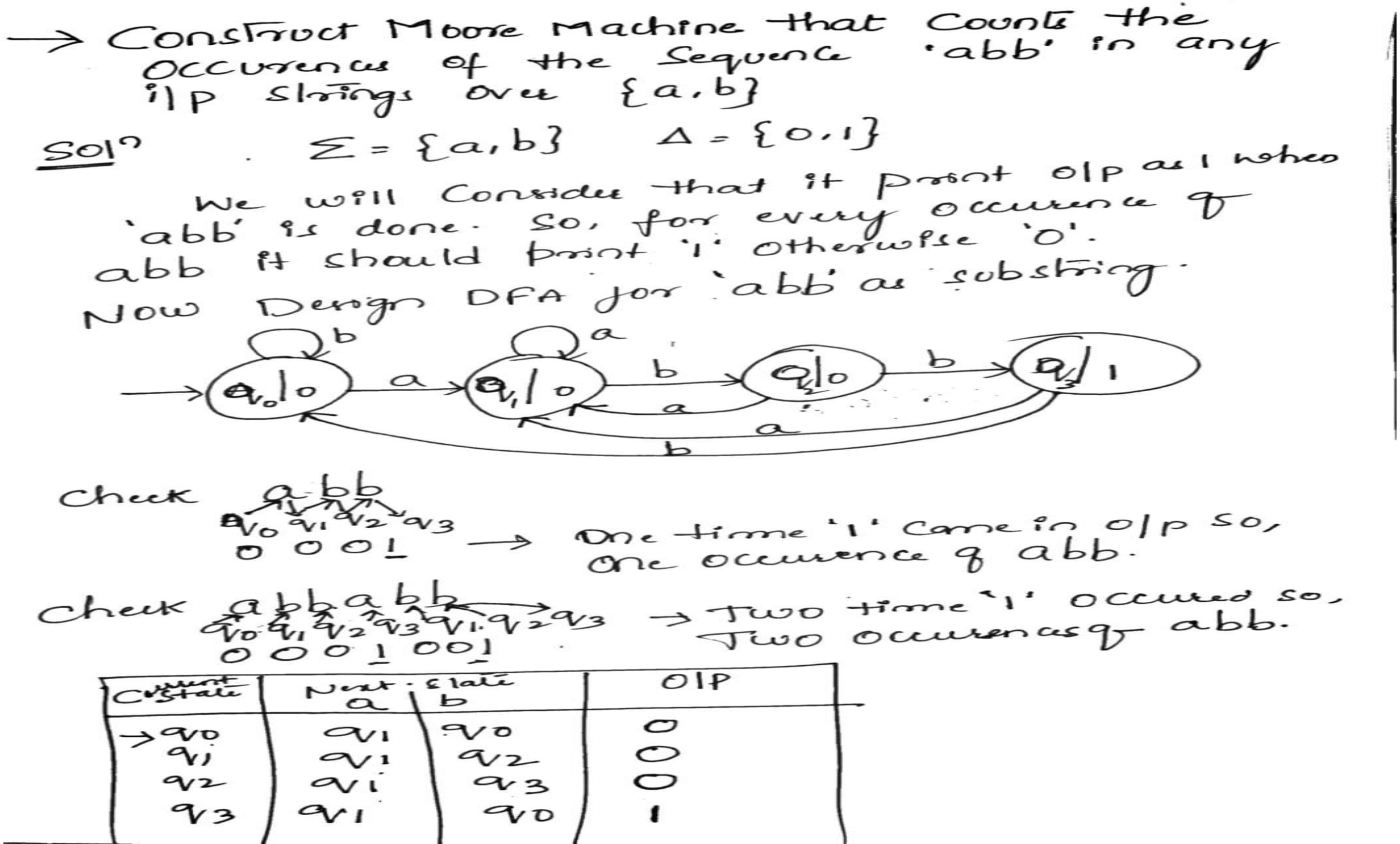
0 1 0 1

q0 q1 q2 q1 q2

b b a b a

Transition Table

| Cu.St | Next State |  | O/P |
|---|---|---|---|
|  | 0 | 1 |  |
| → q0 | q1 | q0 | b |
| q1 | q1 | q2 | b |
| q2 | q1 | q0 | a |



→ Construct Moore Machine that Counts the occurences of the Sequence 'abb' in any i/p strings over {a, b}

Sol^n . $\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

We will Consider that it print o/p as 1 when 'abb' is done. So, for every occurence of abb it should print '1' otherwise '0'.

Now Design DFA for 'abb' as substring.

$q_0/0$
$q_1/0$
$q_2/0$
$q_3/1$
a
b
b
a
b
a
a
b

Check abb

$q_0$ $q_1$ $q_2$ $q_3$

0 0 0 1 → One time '1' Came in o/p so, One occurence of abb.

Check abbabb

$q_0$ $q_1$ $q_2$ $q_3$ $q_1$ $q_2$ $q_3$ → Two time '1' occured so, Two occurences of abb.

0 0 0 1 0 0 1

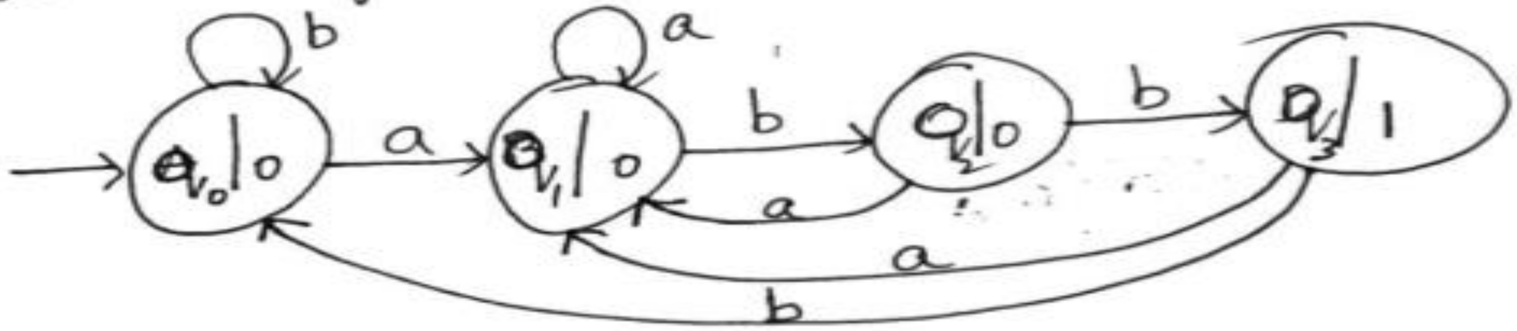
| Current State | Next state a | Next state b | O/P |
|---|---|---|---|
| → $q_0$ | $q_1$ | $q_0$ | 0 |
| $q_1$ | $q_1$ | $q_2$ | 0 |
| $q_2$ | $q_1$ | $q_3$ | 0 |
| $q_3$ | $q_1$ | $q_0$ | 1 |

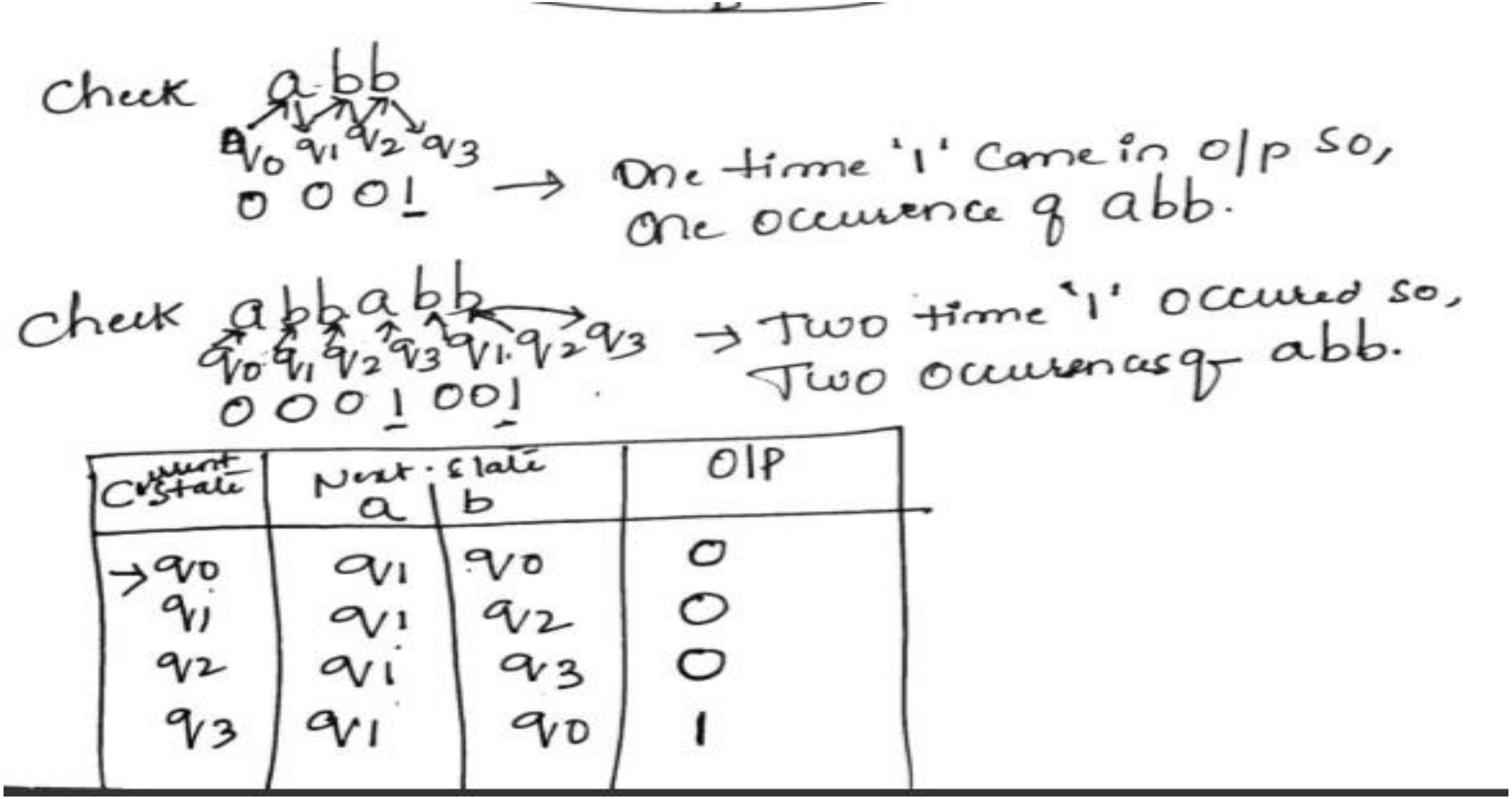
→ Construct Moore Machine that Counts the occurences of the Sequence 'abb' in any i/p strings over $\{a,b\}$

Sol$^n$ . $\Sigma = \{a,b\}$ $\Delta = \{0,1\}$

We will Consider that it print o/p as 1 when 'abb' is done. So, for every occurence of abb it should print '1' otherwise '0'.

Now Design DFA for 'abb' as substring.





Check abb

$q_0$ $q_1$ $q_2$ $q_3$

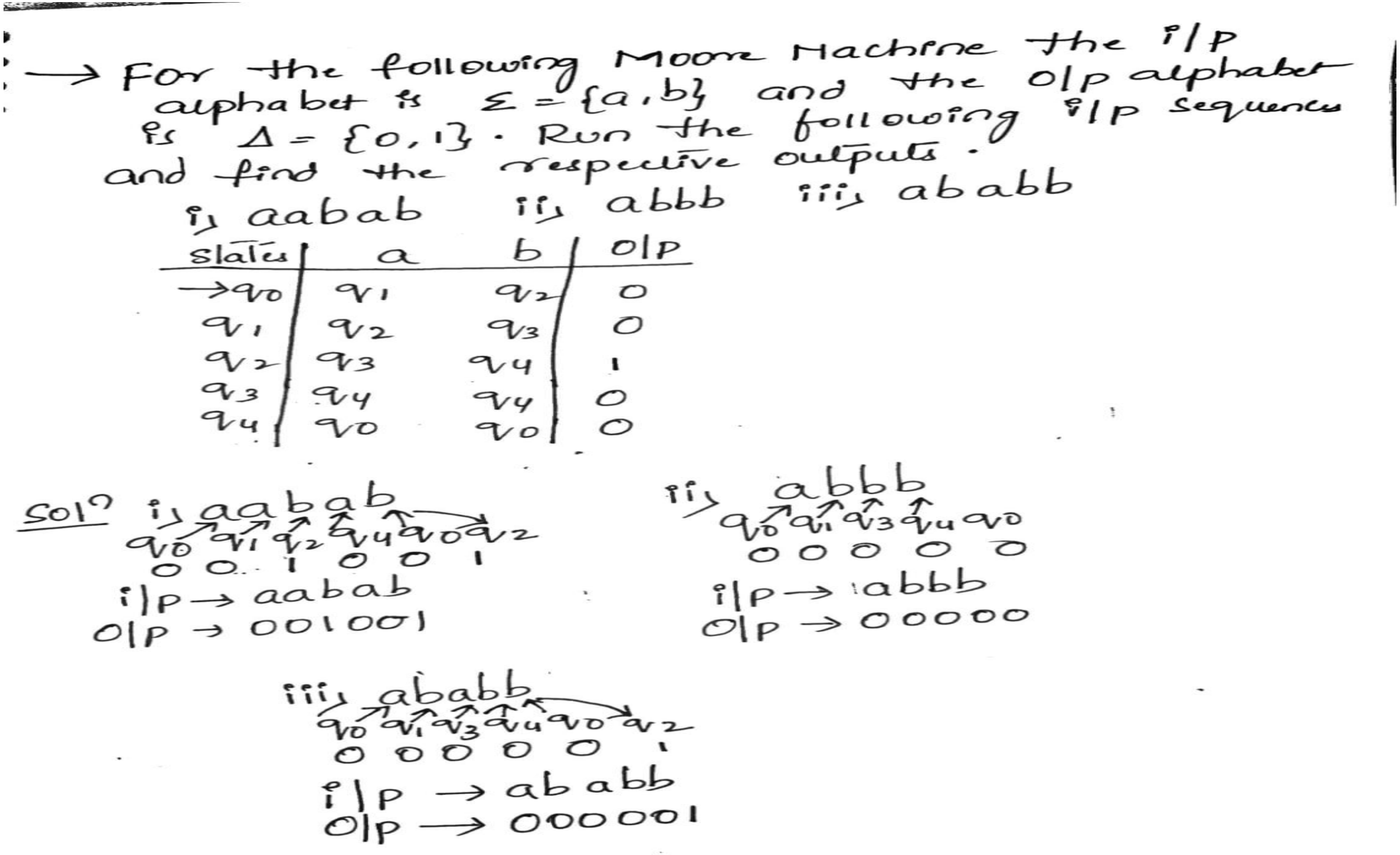
0 0 0 1 → One time '1' came in o/p so, one occurence of abb.

Check abbabb

$q_0$ $q_1$ $q_2$ $q_3$ $q_1$ $q_2$ $q_3$

0 0 0 1 0 0 1 → Two time '1' occured so, Two occurences of abb.

| Current state | Next state a | b | O/P |
|---|---|---|---|
| → $q_0$ | $q_1$ | $q_0$ | 0 |
| $q_1$ | $q_1$ | $q_2$ | 0 |
| $q_2$ | $q_1$ | $q_3$ | 0 |
| $q_3$ | $q_1$ | $q_0$ | 1 |



→ For the following Moore Machine the i/p alphabet is $\Sigma = \{a, b\}$ and the o/p alphabet is $\Delta = \{0, 1\}$. Run the following i/p sequences and find the respective outputs.

i) aabab  ii) abbb  iii) ababb

| states | a | b | o/p |
|---|---|---|---|
| $\rightarrow q_0$ | $q_1$ | $q_2$ | 0 |
| $q_1$ | $q_2$ | $q_3$ | 0 |
| $q_2$ | $q_3$ | $q_4$ | 1 |
| $q_3$ | $q_4$ | $q_4$ | 0 |
| $q_4$ | $q_0$ | $q_0$ | 0 |

Sol^n i) aabab

$q_0 \; q_1 \; q_2 \; q_4 \; q_0 \; q_2$

0 0 1 0 0 1

i/p → aabab

o/p → 001001

ii) abbb

$q_0 \; q_1 \; q_3 \; q_4 \; q_0$

0 0 0 0 0

i/p → abbb

o/p → 00000

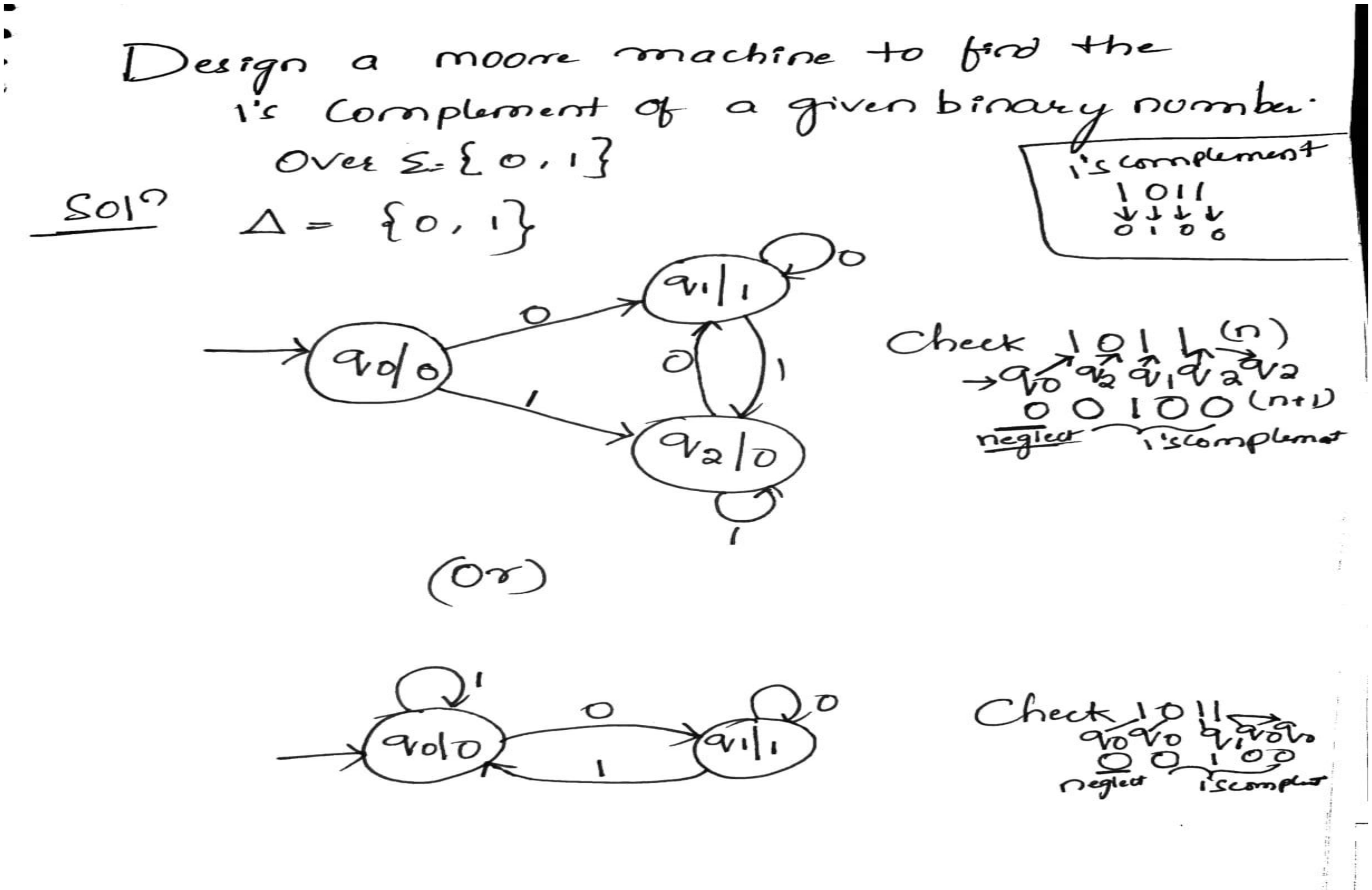
iii) ababb

$q_0 \; q_1 \; q_3 \; q_4 \; q_0 \; q_2$

0 0 0 0 0 1

i/p → ababb

o/p → 000001



Design a moore machine to find the 1's Complement of a given binary number. Over $\Sigma = \{0, 1\}$

Sol$^n$ $\Delta = \{0, 1\}$

1's complement

1011

↓↓↓↓

0100

$q_0/0$
$q_1/1$
$q_2/0$
0
0
0
1
1
1

Check $1\ 0\ 1\ 1$ (n)

$\rightarrow q_0\ q_2\ q_1\ q_2\ q_2$

$0\ \underbrace{0\ 1\ 0\ 0}_{\text{1's complemet}}$ (n+1)

neglect

(Or)

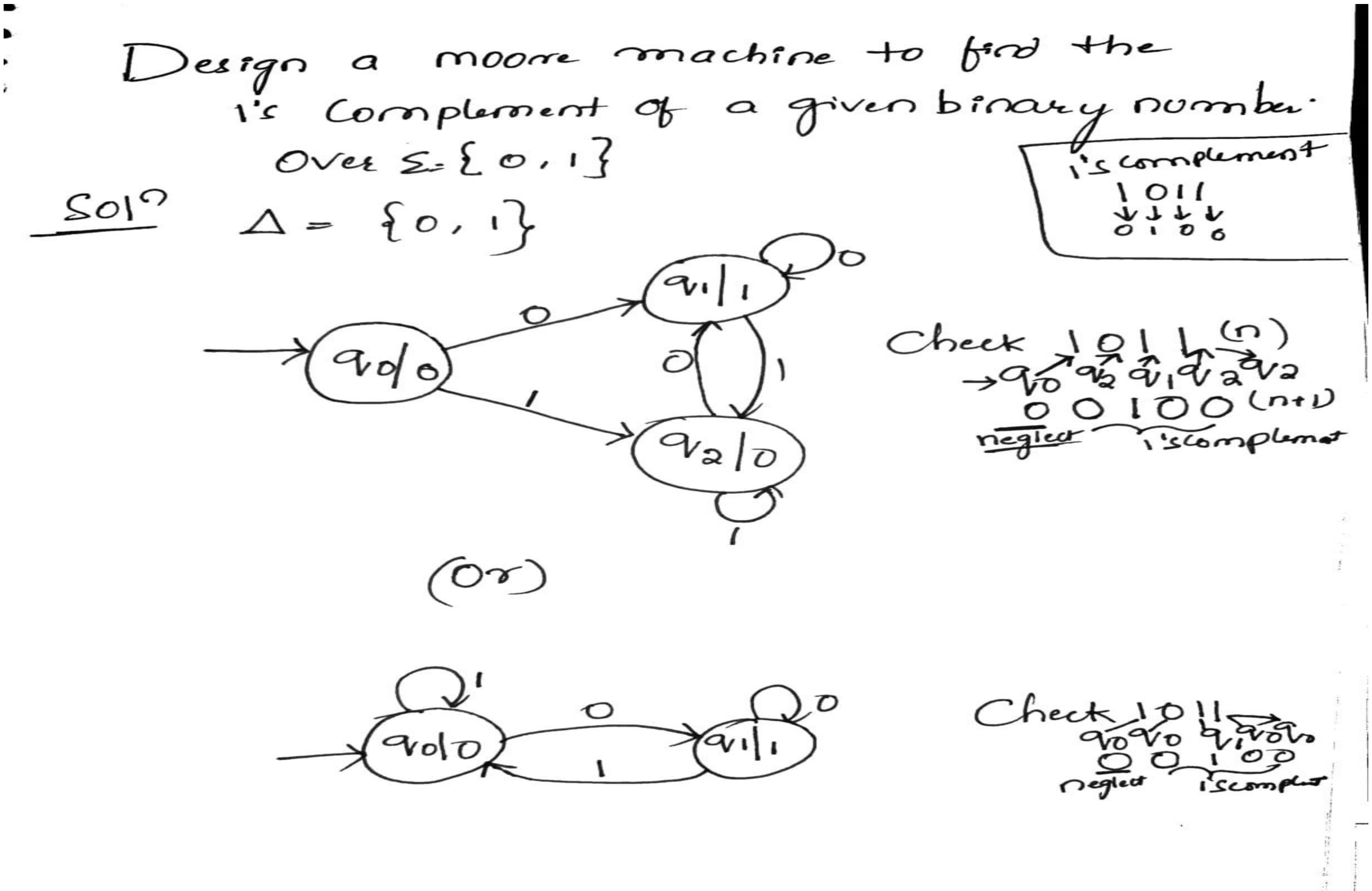
$q_0/0$
$q_1/1$
1
0
0
1

Check $1\ 0\ 1\ 1$

$q_0\ q_0\ q_0\ q_1\ q_0\ q_0$

$0\ \underbrace{0\ 1\ 0\ 0}_{\text{1'scomplmt}}$

neglect



Design a moore machine to find the 1's Complement of a given binary number. Over Σ = {0, 1}

Soln $\Delta = \{0, 1\}$

1's complement

1011

↓↓↓↓

0100

Check 1011 (n)

→ q0 q2 q1 q2 q2

0 0 1 0 0 (n+1)

neglect 1's complement

(Or)

Check 1011

q0 q0 q1 q0 q0

0 0 1 0 0

neglect 1's compl

# Mealy Machine

- A **Mealy machine** is a finite state machine that has an output value rather than a final state.

- For a given input, the machine generates a corresponding output.

- The output of the Mealy machine depends on the present state of the FA as well as the current input symbol.

- Unlike other finite automata that determine the acceptance of a particular string in a given language, Mealy machines determine the output against the given input.

# Formal Definition Of Mealy Machine

The Mealy machine is a 6 tuple machine ($Q$,Σ,$q$0,Δ,$\delta$,$\lambda$):

$Q$ : This is a set of states.

Σ: This is a set of input alphabets.

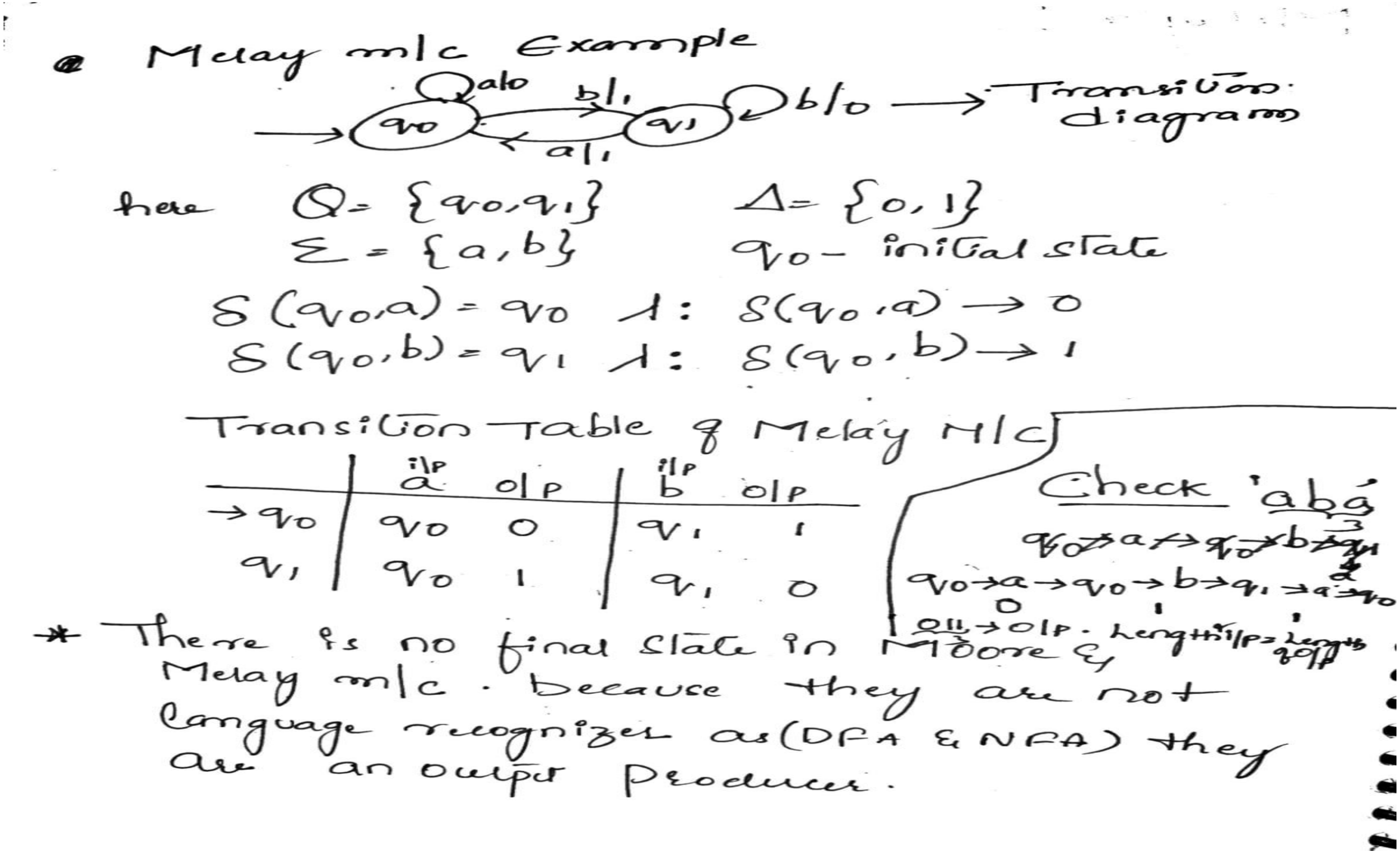
$q$0: This is an initial state.

Δ: This is a set of output states

$\delta$: This is a transition function, that is: $Q \times \Sigma \rightarrow Q$

$\lambda$: This is an output function, that is: $Q \times \Sigma \rightarrow \Delta$

**Note:** The output function means that for every transition at a particular state, there is a corresponding output associated with it.



a|0
b|1
b|0
q0
q1
a|1

# Melay m|c Example

→ Transition diagram

here $Q = \{q_0, q_1\}$  $\Delta = \{0, 1\}$

$\Sigma = \{a, b\}$  $q_0$ – initial state

$\delta(q_0, a) = q_0$  $\lambda: \delta(q_0, a) \rightarrow 0$

$\delta(q_0, b) = q_1$  $\lambda: \delta(q_0, b) \rightarrow 1$

Transition Table of Melay M|C

|  | i/p a | o/p | i/p b | o/p |
|---|---|---|---|---|
| → $q_0$ | $q_0$ | 0 | $q_1$ | 1 |
| $q_1$ | $q_0$ | 1 | $q_1$ | 0 |

Check 'aba'

~~$q_0 \rightarrow a \rightarrow q_0 \rightarrow b \rightarrow q_1$~~

$q_0 \rightarrow a \rightarrow q_0 \rightarrow b \rightarrow q_1 \rightarrow a \rightarrow q_0$

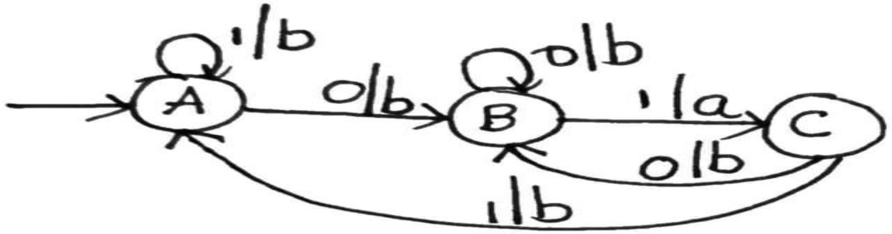
0 1 1

011 → o/p. Length of i/p = Length of o/p

* There is no final state in Moore & Melay m/c. Because they are not language recognizer as (DFA & NFA) they are an output Producer.

1) Construct a Melay machine that prints 'a' whenever the sequence '01' is encountered in any input binary string.

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$



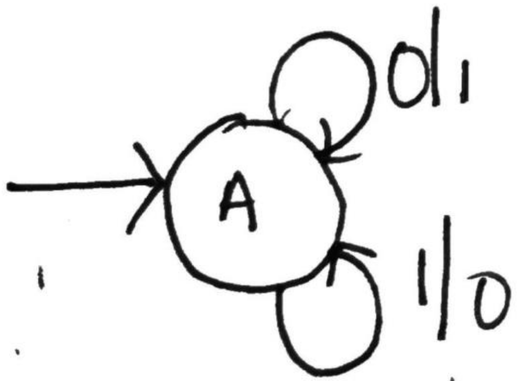
Check string 0110 length of i/p = length of o/p

b$\underline{a}$bb

Check string 10001

bbbb$\underline{a}$

Constrruct a Melay machine that produces the 1's complement of any binary i/p String.



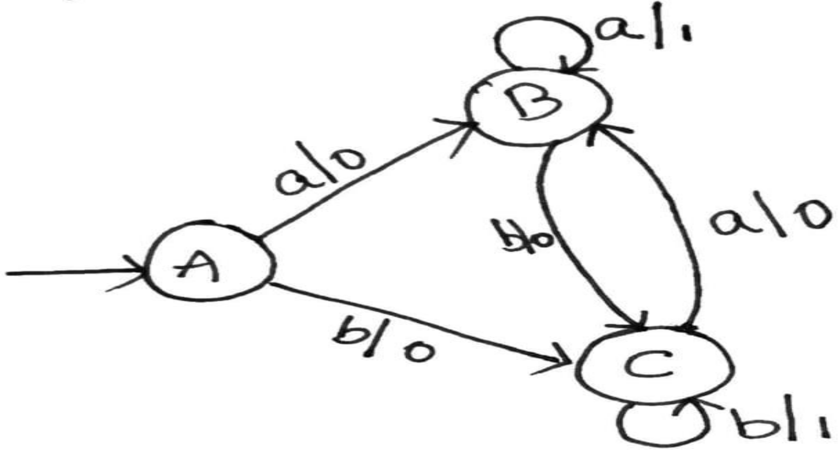
10100
01011

|  | 0 | o/p | 1 | o/p |
|---|---|---|---|---|
| → Aq0 | q0 | 1 | q0 | 0 |

Design a Melay m/c accepting the language consisting of strings from Σ*, where Σ = {a,b} and the strings should end with aa or bb

Sol^n

print '1' as o/p whenever we see aa – 1
bb – 1

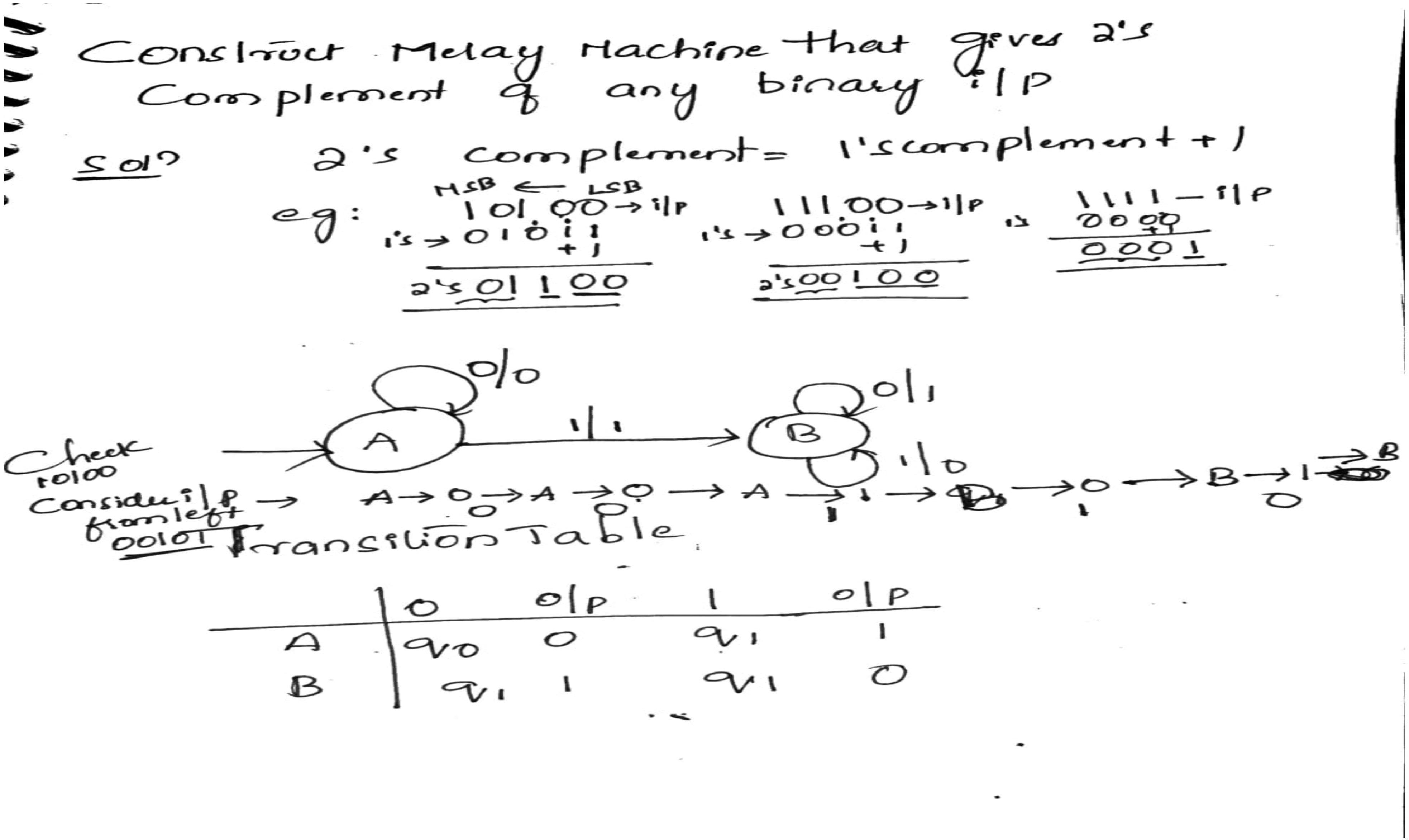


a$\underline{bb}$ →
b$\underline{aa}$

|  | a | o/p | b | o/p |
|---|---|---|---|---|
| A | B | 0 | C | 0 |
| B | B | 1 | C | 0 |
| C | B | 0 | C | 1 |

ba
00 → NO '1'

A → aa
0 $\underline{1}$ → accep ✓



Construct Melay Machine that gives 2's Complement of any binary i/p

Sol: 2's complement = 1's complement + 1

eg: MSB ← LSB

10100 → i/p | 11100 → i/p | 1111 – i/p
---|---|---
1's → 01011 | 1's → 00011 | 1's 0000
+1 | +1 | +1
2's 01100 | 2's 00100 | 0001

0/0
1/1
0/1
1/0
A
B

Check 10100

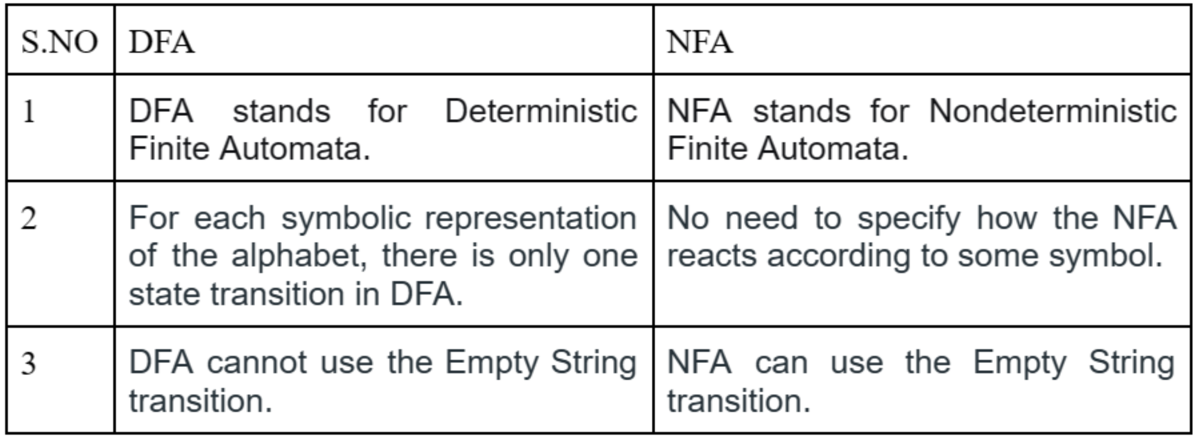
Consider i/p from left 00101 → A → 0 → A → 0 → A → 1 → B → 0 → B → 1 → B

output: 0 0 1 1 0

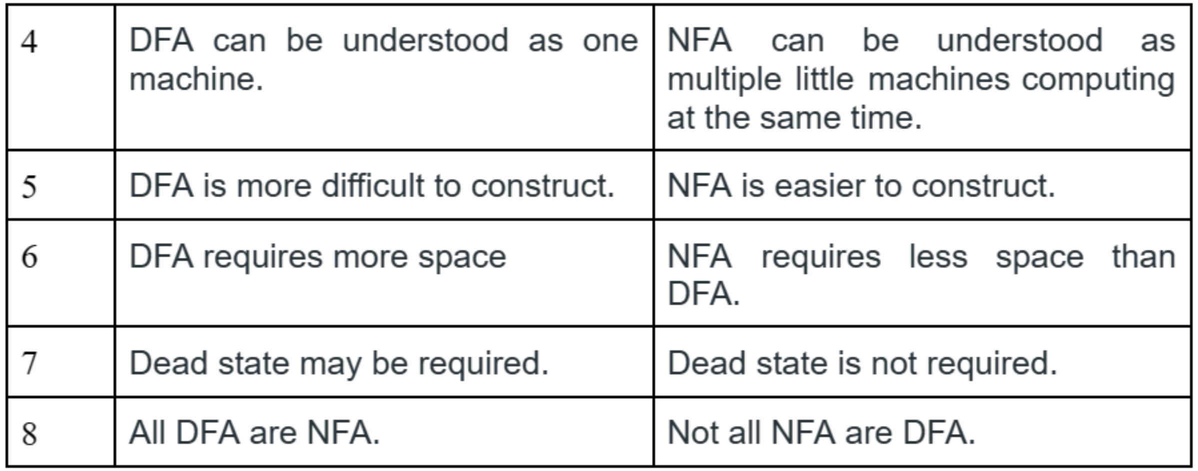
Transition Table

| | 0 | o/p | 1 | o/p |
|---|---|---|---|---|
| A | q0 | 0 | q1 | 1 |
| B | q1 | 1 | q1 | 0 |

# Differences between DFA and NFA



| S.NO | DFA | NFA |
|---|---|---|
| 1 | DFA stands for Deterministic Finite Automata. | NFA stands for Nondeterministic Finite Automata. |
| 2 | For each symbolic representation of the alphabet, there is only one state transition in DFA. | No need to specify how the NFA reacts according to some symbol. |
| 3 | DFA cannot use the Empty String transition. | NFA can use the Empty String transition. |



| | | |
|---|---|---|
| 4 | DFA can be understood as one machine. | NFA can be understood as multiple little machines computing at the same time. |
| 5 | DFA is more difficult to construct. | NFA is easier to construct. |
| 6 | DFA requires more space | NFA requires less space than DFA. |
| 7 | Dead state may be required. | Dead state is not required. |
| 8 | All DFA are NFA. | Not all NFA are DFA. |