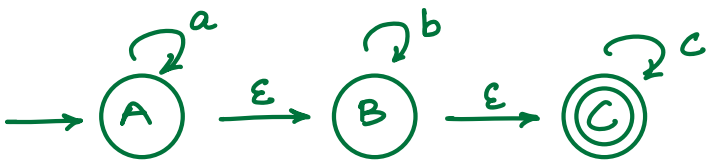
Conversion ε-NFA to NFA

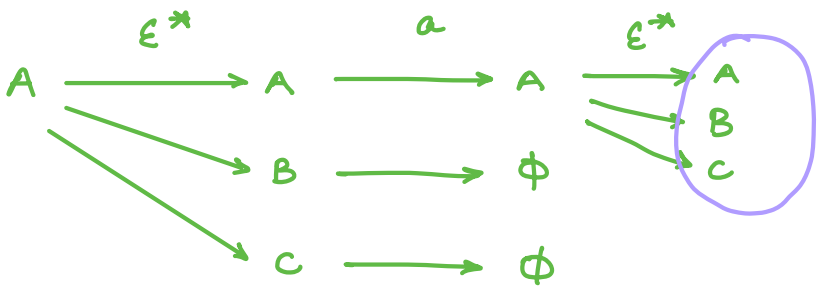
eg:



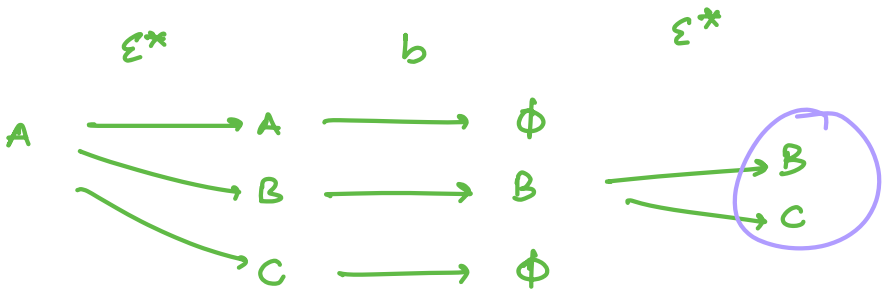
$\Sigma = \{a, b, c\}$

ε-closure (A) = ABC
ε-closure (B) = BC
ε-closure (C) = C

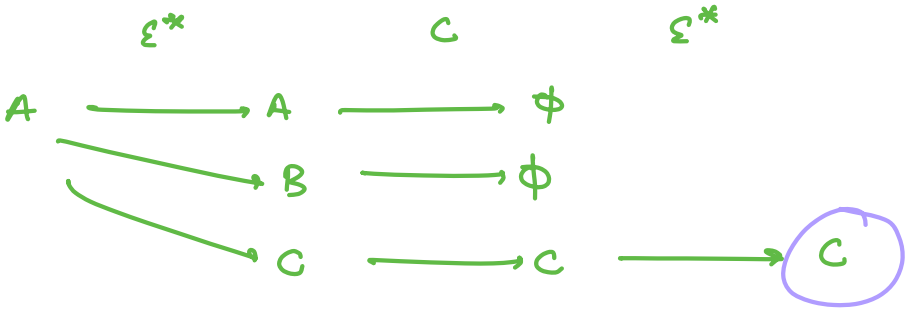
A, a



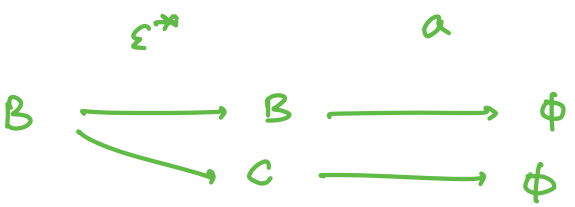
A, b



A, C

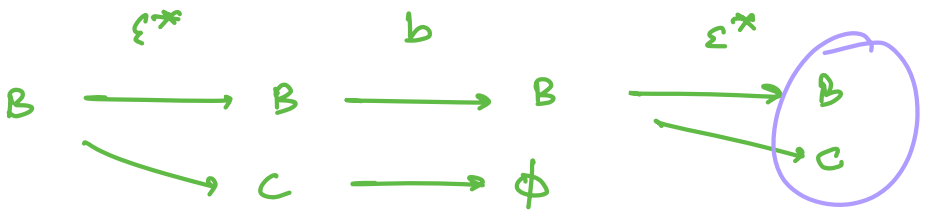


B, a

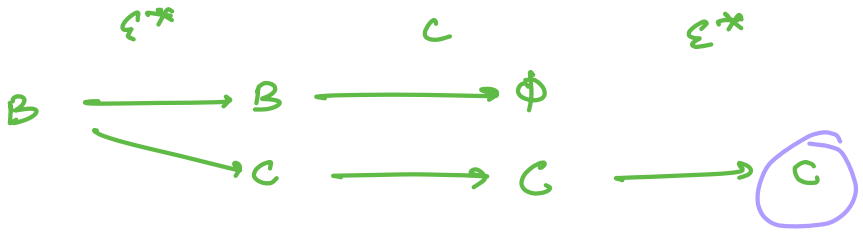


find NFA (B, a) no transition

B,b



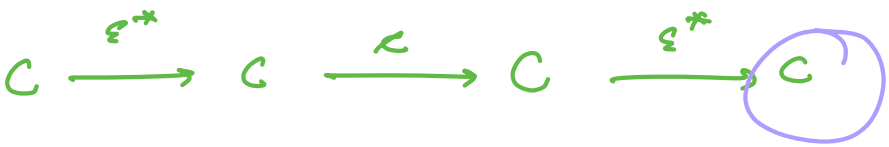
B,C



C,a

C,b

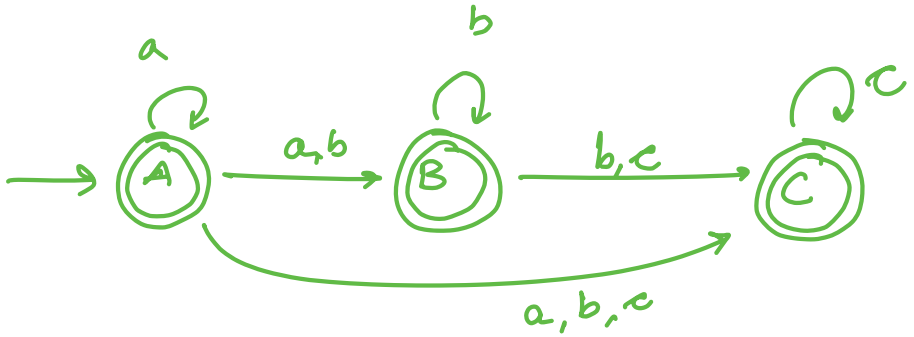
C,c



State Transition Table (NFA)

	a	b	c
A	ABC	BC	C
B	ϕ	BC	C
C	ϕ	ϕ	C

State Transition Diagram (NFA)



ε-NFA NFA DFA

Equal in Power
(work that can be done by one can also be done by other)

---

finite Automata

$(Q, \Sigma, \delta, q_0, F)$

- without output
  - DFA: $\delta: Q \times \Sigma \to Q$
  - NFA: $\delta: Q \times \Sigma \to 2^Q$
  - ε-NFA: $Q \times (\Sigma \cup \varepsilon) \to 2^Q$

  Equivalent in power
- With output
  - Moore
  - Mealy

  Equivalent in power.

$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Both moore & mealy m/c is deterministic

Q: finite set of states

Σ: input alphabet

δ: transition fn

$Q \times \Sigma \to Q$

q_0 : initial state

Δ : output alphabet (Symbols that will be printed)

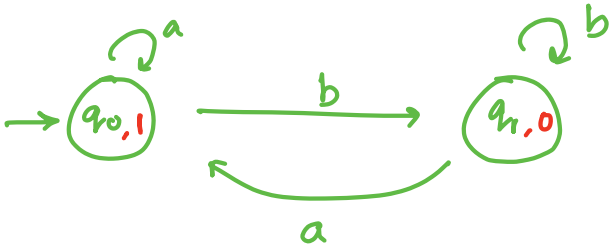
λ : output function

→ only difference between moore and mealy machine.

→ no ambiguity

FA with output (Deterministic)

Moore



- Output is associated with state

- eg: q_0, output is 1
  q_1, output is 0

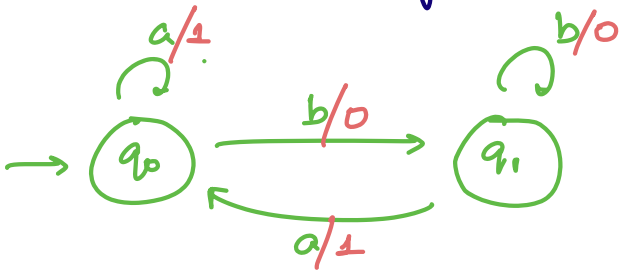
- eg: Input: ab

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$

1 1 0

output: 110

Mealy



- Output is associated with current state and input alphabet.

- eg: q_0, a → 1
  q_0, b → 0
  q_1, a → 1
  q_1, b → 0

- eg: Input: ab

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$

1 0

output: 10

⊙ Input: length n
Output: length n+1

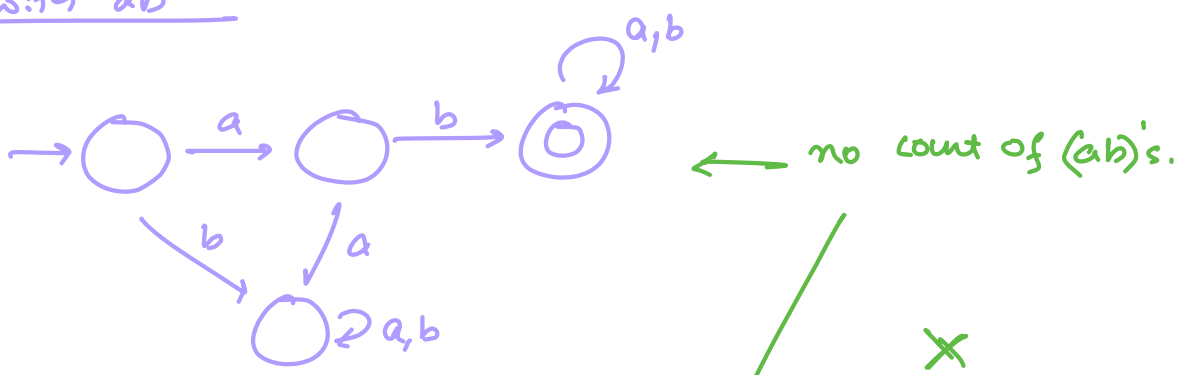
⊙ Input: length n
Output: length n

Q: Construct a moore m/c that takes set of all strings over {a,b} as input and prints '1' as o/p for every occurrence of 'ab' as a substring.

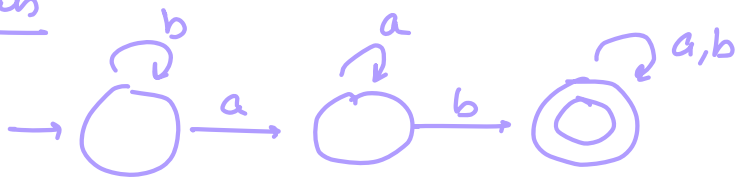
3 options:

- starting with ab
- substring ab
- ending with ab

Starting with ab

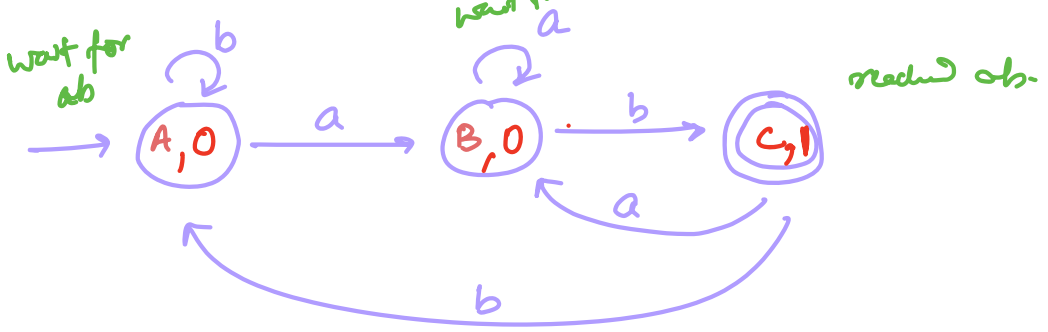


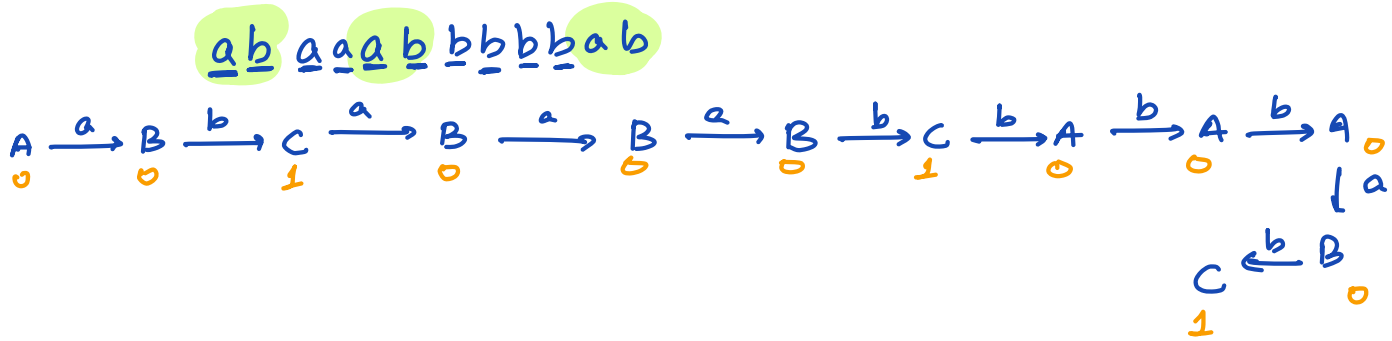
Substring ab



Ending with ab

count (ab)



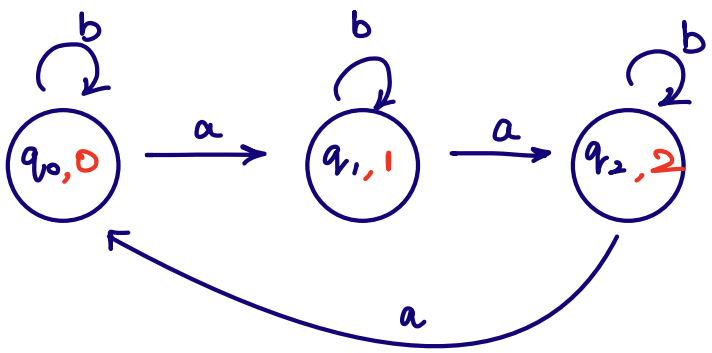


Moore ⇄ Mealy

Moore and mealy are equivalent in power.

Convert Moore to mealy machine:

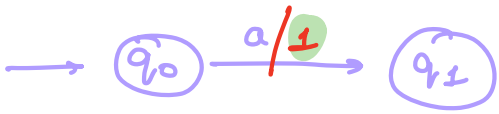
Eg:

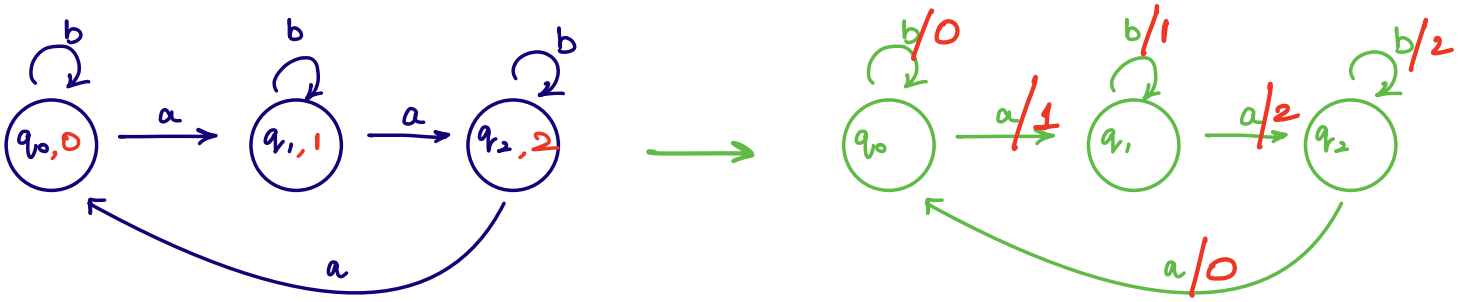


count a % 3

output is associated with every state, move output to every transition.

$(q_0, a) \rightarrow q_1$ and on q_1 output is 1





State Transition Table

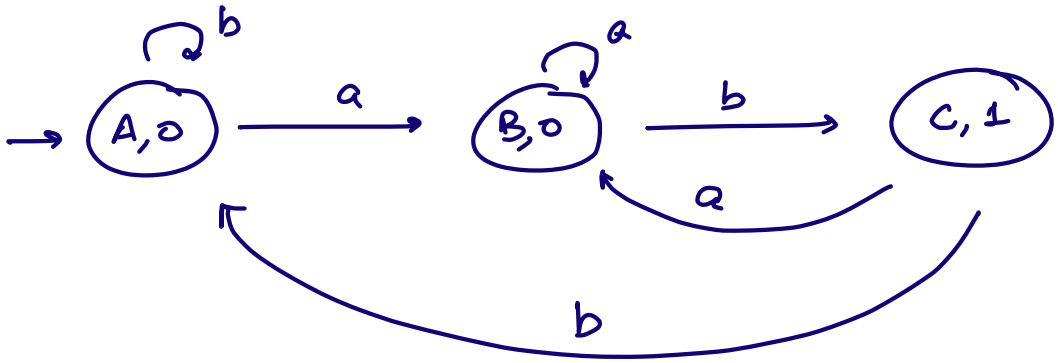
moore m/c

	a	b	Δ
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

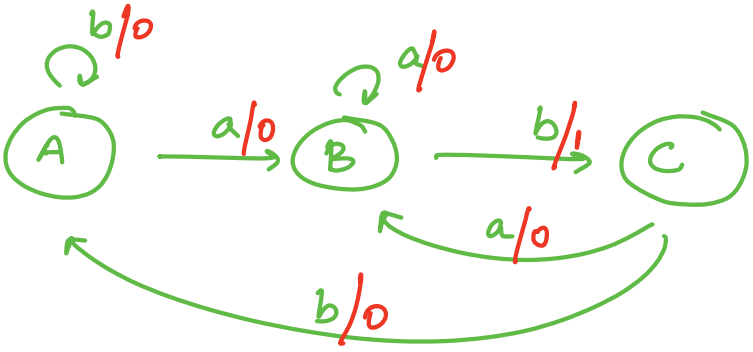
mealy m/c:

	a	b
q_0	$q_1/1$	$q_0/0$
q_1	$q_2/2$	$q_1/1$
q_2	$q_0/0$	$q_2/2$

Eg:



Mealy machine:



State Transition Table:

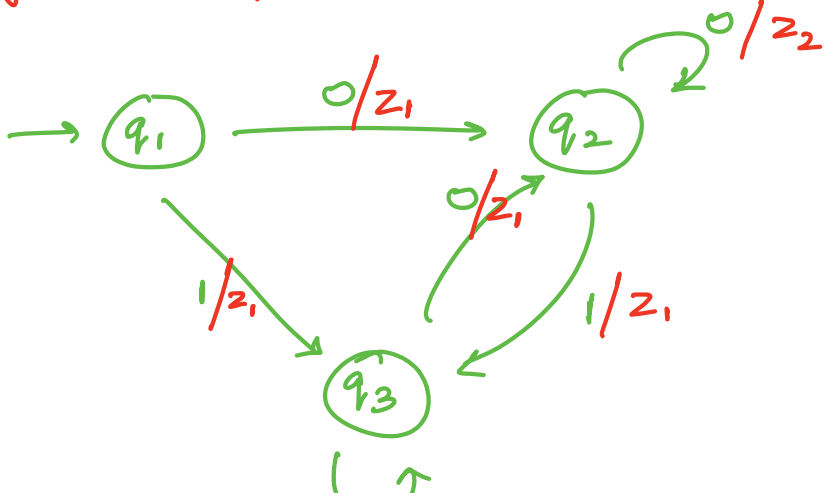
Moore:

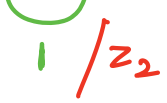
	a	b	Δ
A	B	A	0
B	B	C	0
C	B	A	1

Mealy:

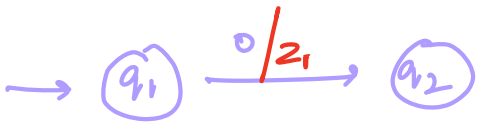
	a	b
A	B/0	A/0
B	B/0	C/1
C	B/0	A/0

Convert mealy to Moore machine:

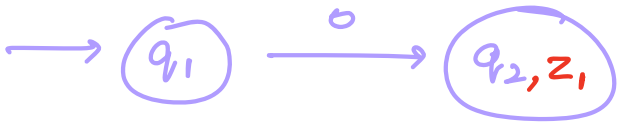


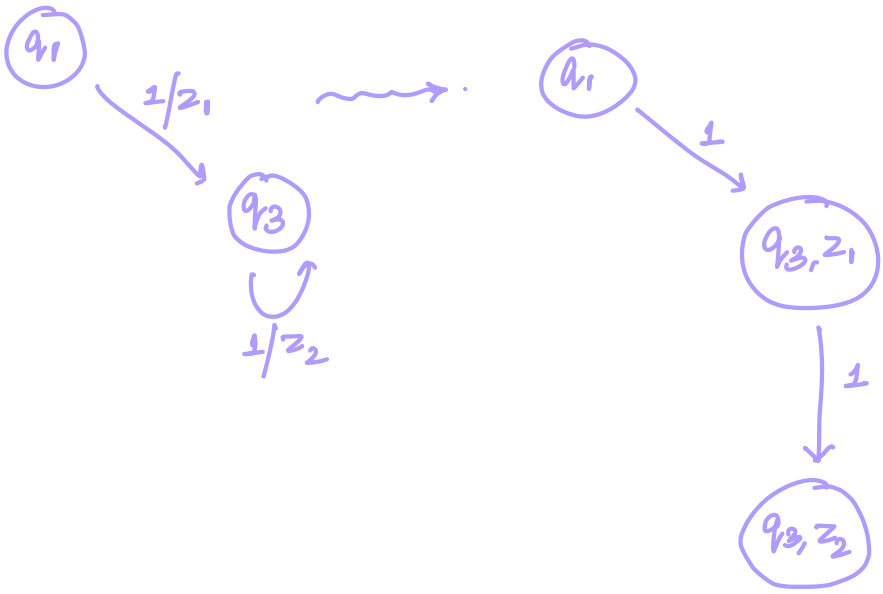


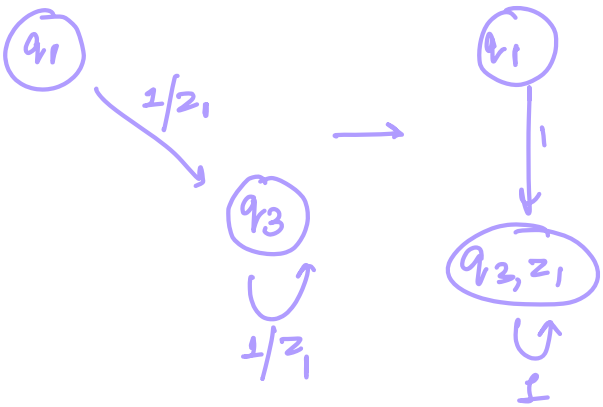
q_1 on 0 is going to q_2 & o/p is z_1



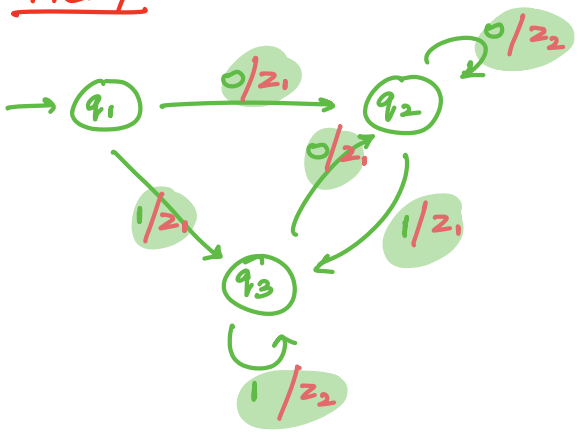
move the o/p from transition to state

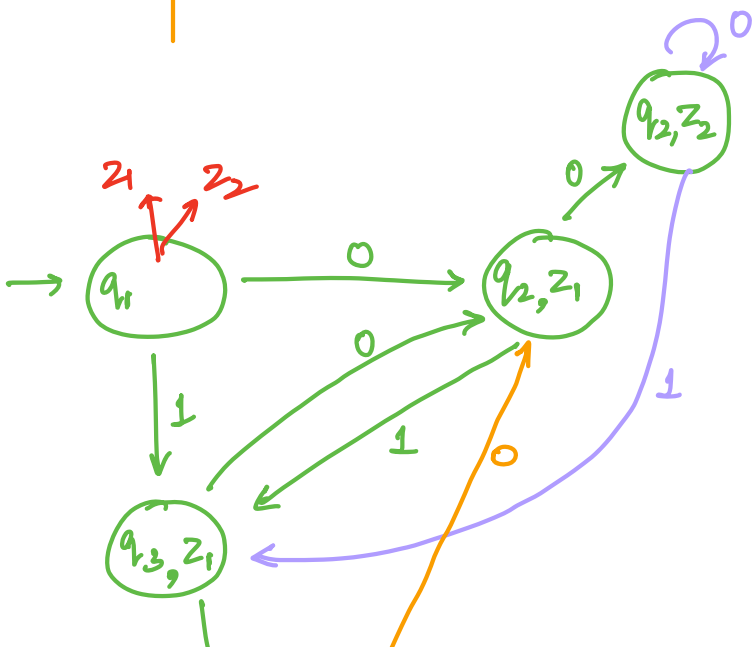






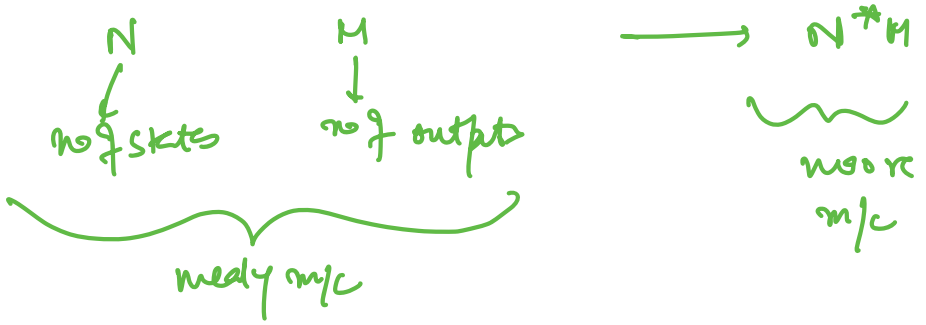
mealy





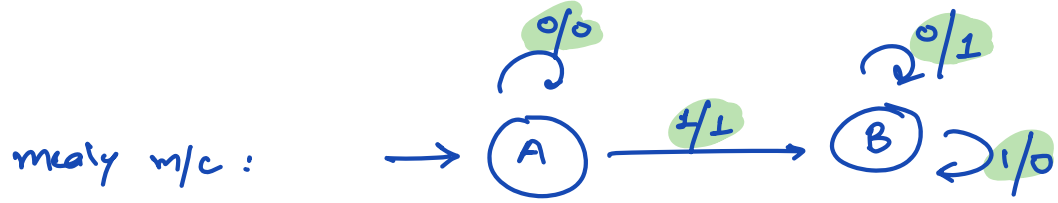
moore ⟶ mealy : no. of states remain same

mealy ⟶ moore : no of states might increase

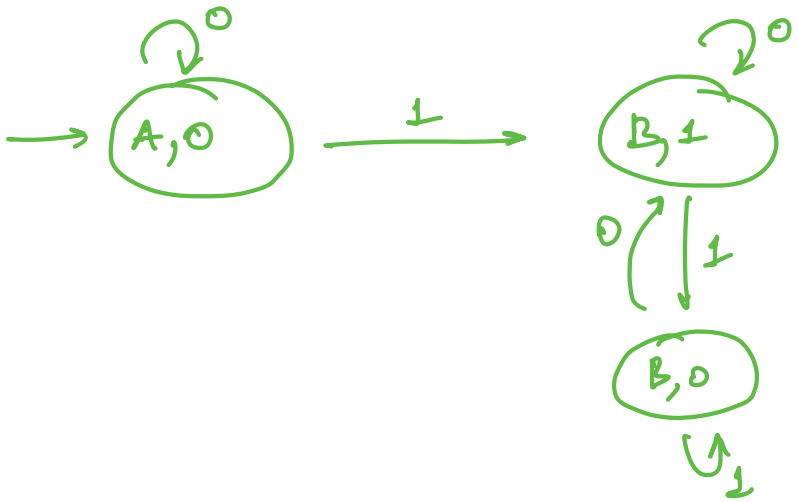


q_1 didnot get any output bcz there is no incoming edge to q_1. Keep either z_1 or z_2 with q_1.

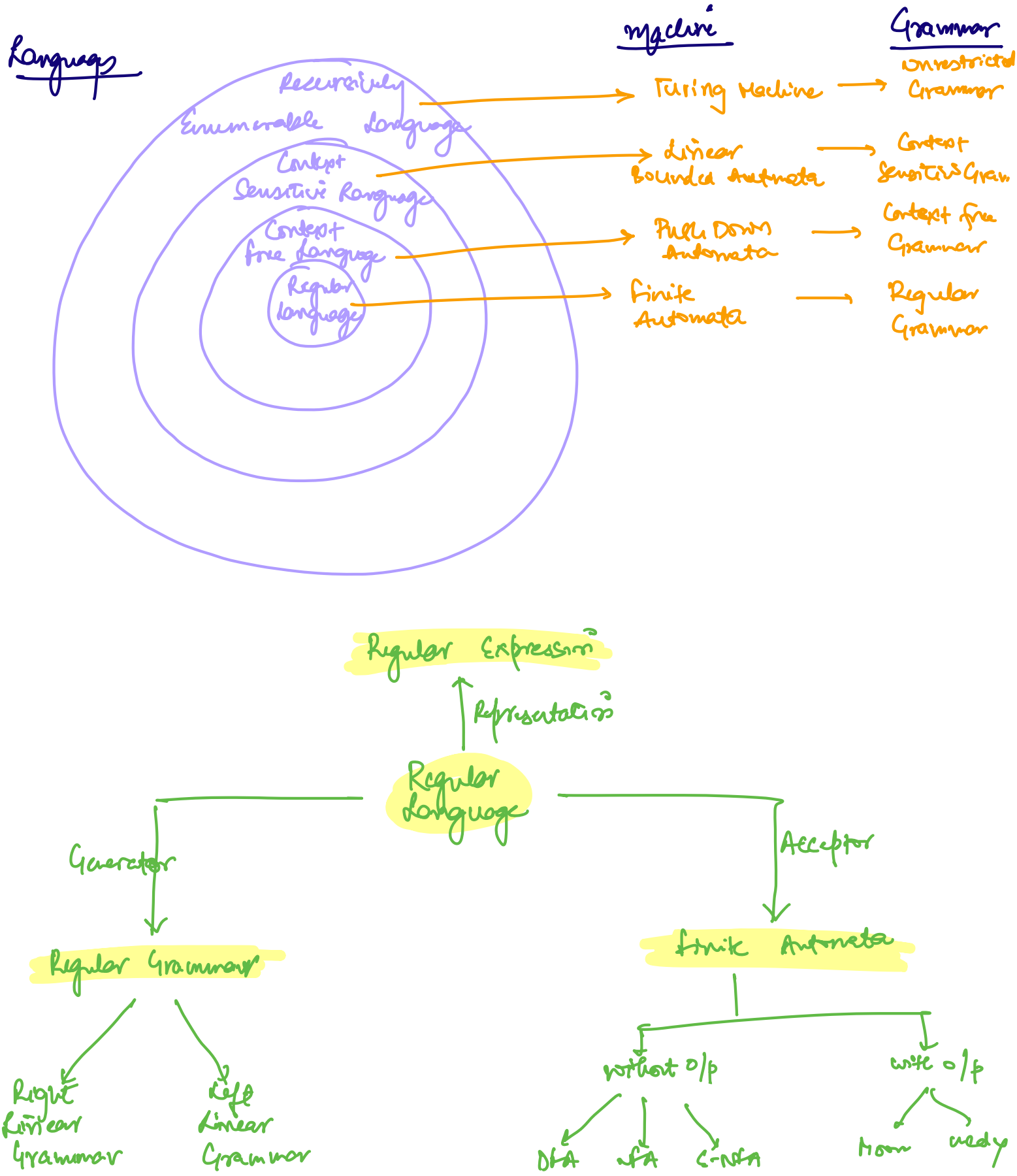
Eg:



moore m/c



Language which is accepted by finite automata is called as REGULAR LANGUAGE.


Language
Machine
Grammar
Recursively Enumerable Language
Context Sensitive Language
Context free Language
Regular language
Turing Machine
Unrestricted Grammar
Linear Bounded Automata
Context Sensitive Gram.
Push Down Automata
Context free Grammar
Finite Automata
Regular Grammar
Regular Expression
Representation
Regular Language
Generator
Acceptor
Regular Grammar
Finite Automata
Right Linear Grammar
Left Linear Grammar
without o/p
with o/p
DFA
NFA
ε-NFA
Moore
Mealy