Deterministic Finite Automata (DFA)

→ a finite state machine that accepts a given string of symbols, running through a sequence uniquely determined by the string.

DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ consisting of

- $Q \rightarrow$ finite set of states
- $\Sigma \rightarrow$ finite set of input symbols
- $\delta \rightarrow$ a transition function $Q \times \Sigma \rightarrow Q$
- $q_0 \rightarrow$ initial or start state; $q_0 \in Q$
- $F \rightarrow$ Final or accepting state; $F \subseteq Q$

start state represented by $\rightarrow q_0$

Final/Accepting state represented by F

In transition table

	Q	Σ
Start state	$\rightarrow q_0$	
Final/Accepting state	$* q_1$	
	$* q_2$	

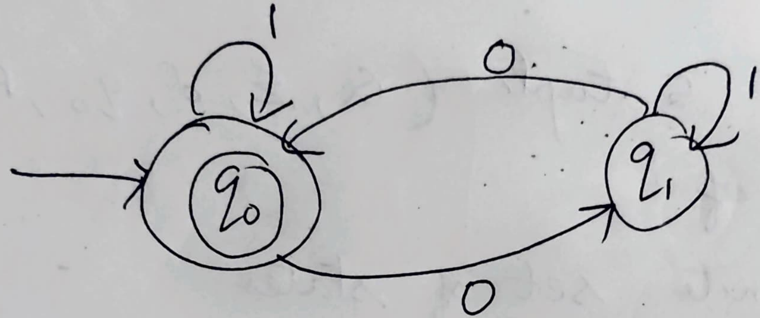
OR

	Q	Σ
Start state	$\rightarrow q_0$	
Final/Accepting state	q_1	
	q_2	

DFA

Examples

① Construct DFA, M with a binary alphabet $\Sigma = \{0,1\}$, which requires input contains an even number of 0's.



$M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1\}$

$\Sigma = \{0,1\}$

$q_0 = q_0$

$F = q_1$

$\delta \rightarrow$ state transition table

$Q \backslash \Sigma$	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_1

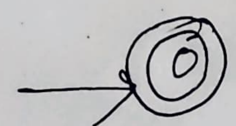
(2) Construct DFA for L = { even number of a's and even number of b's }

string's = { ε, aa, bb, aaaa, bbbb, aabb, abab, baab, baba, abba, abba, bbaa, etc. }

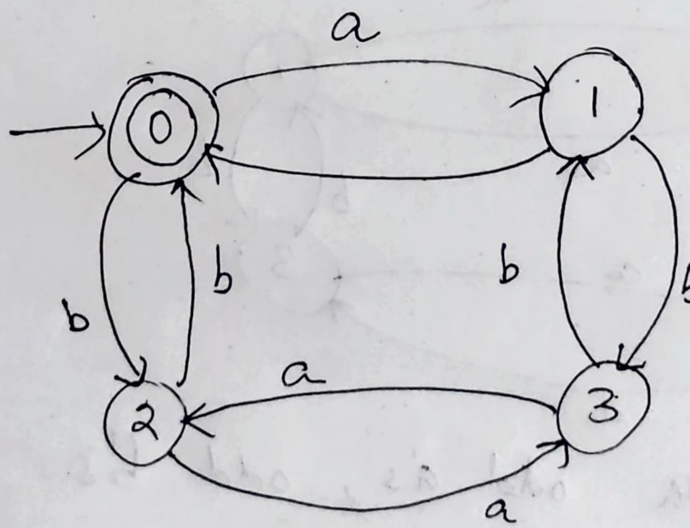
Solution

Step 1 :-

0 number of a, 0 number of b

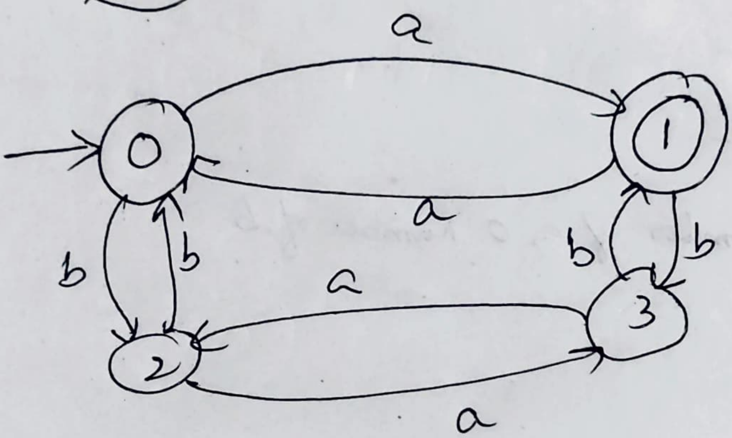


step 2 :-

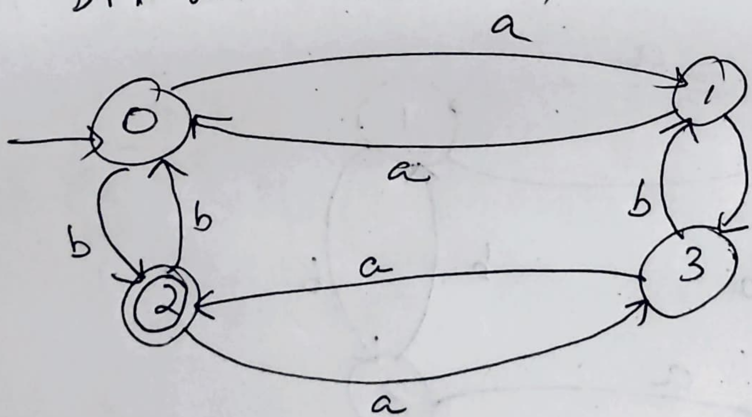


(3) Construct DFA which accepts odd number of a's and even no. of b's.

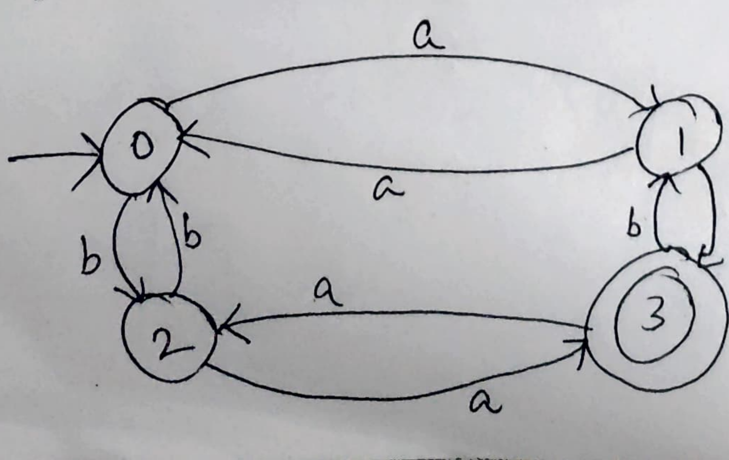
Solution



(4) DFA with odd b's, even a's



(5) DFA with odd a's, odd b's.



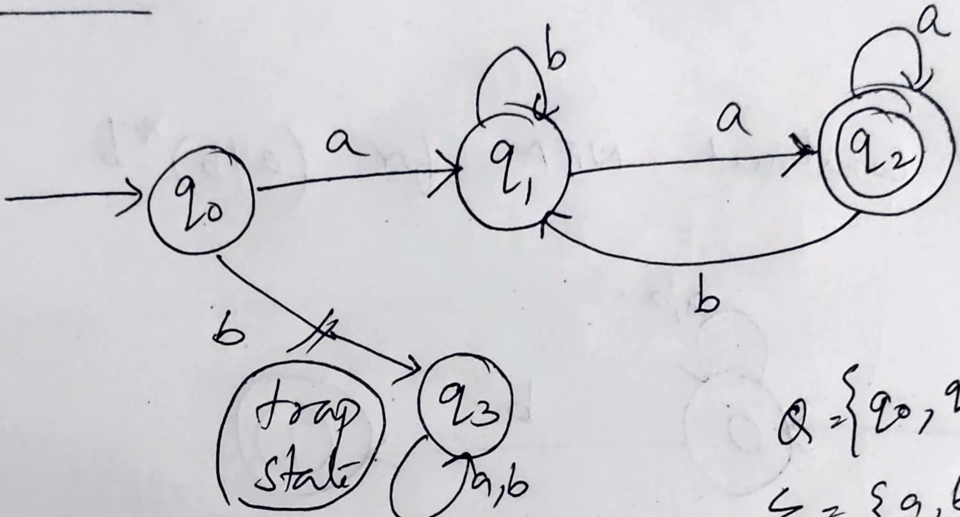
(6) DFA which accepts language $L = a\underline{w}a$

$w = (a/b)^*$

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

Language $= a(a/b)^* a$

Solution



δ		a	b
	q_0	q_1	trapstate
	q_1	q_2	q_1
	q_2	q_2	q_1

$Q = \{q_0, q_1, q_2\}$

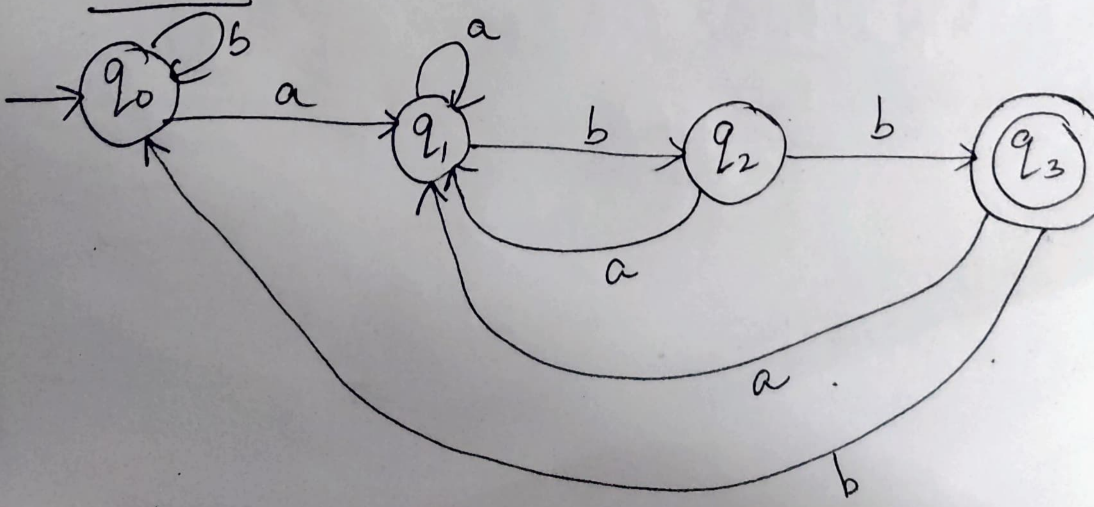
$\Sigma = \{a, b\}$

$q_0 = q_0$

$F = q_2$

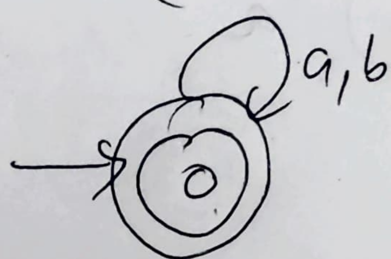
(7) DFA with language $L = (a/b)^* abb$

Solution

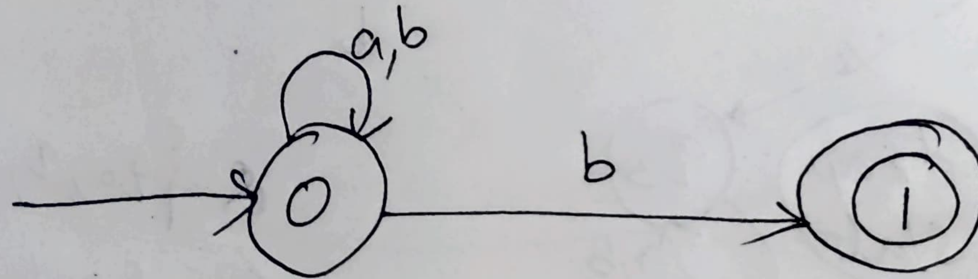


1. Construct NFA for

$(a|b)^*$

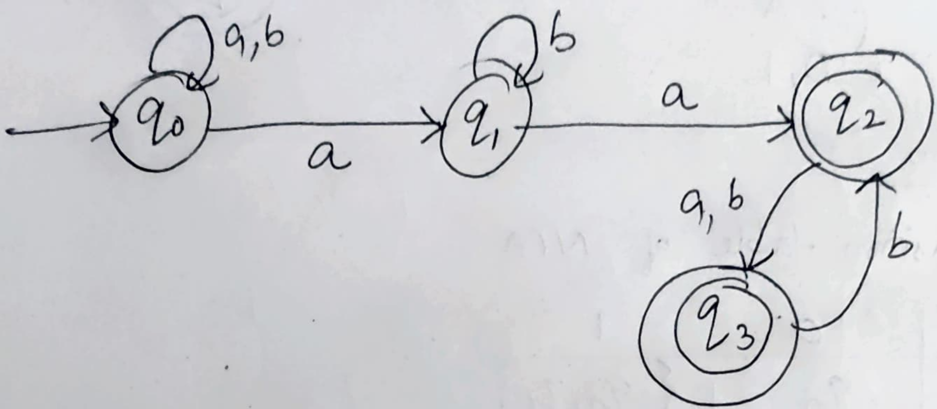


2. construct NFA for $(a|b)^* b$



NFA to DFA Conversion

① Convert the following NFA to DFA



NFA → with same input symbol can reach multiple state

Solution

NFA Transition table

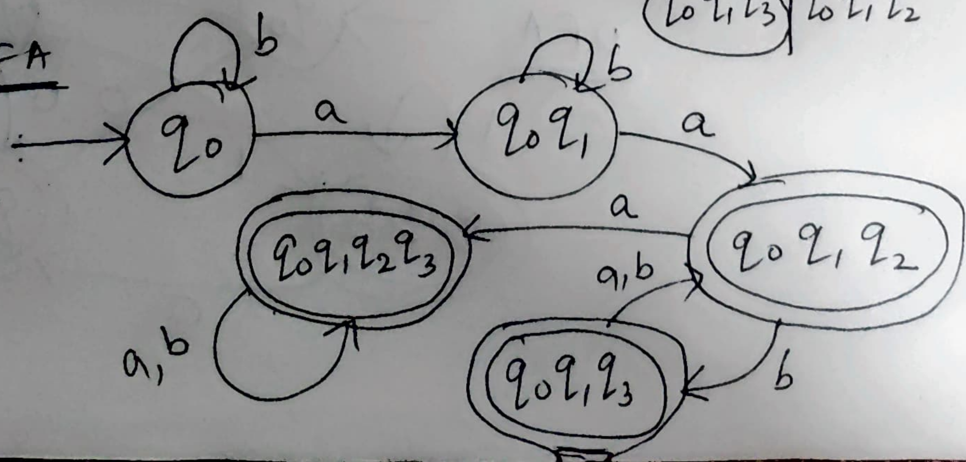
Q \ Σ	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3	—	q_2

⟹

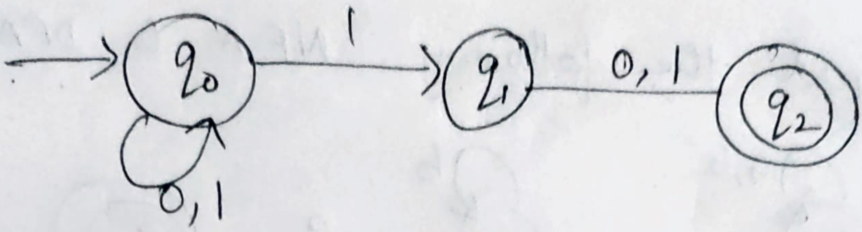
DFA transition table

	a	b
→ q_0	q_0q_1	q_0
q_0q_1	$q_0q_1q_2$	q_0q_1
$q_0q_1q_2$	$q_0q_1q_2q_3$	$q_0q_1q_3$
$q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_0q_1q_2q_3$
$q_0q_1q_3$	$q_0q_1q_2$	$q_0q_1q_2$

DFA



② Convert the following NFA to DFA



<u>Solution</u>

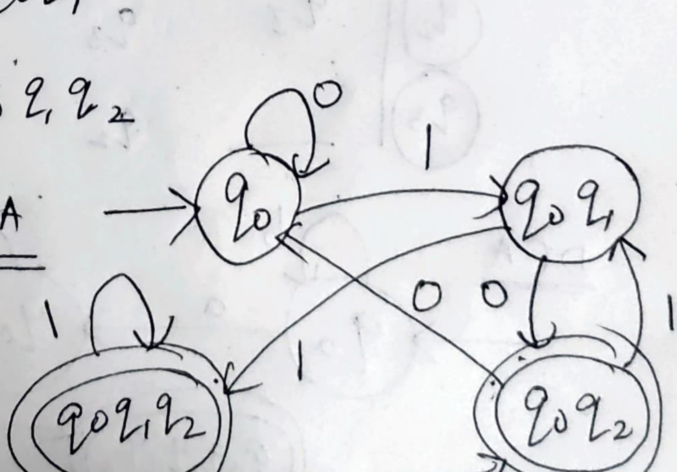
Transition table of NFA

$Q \setminus \Sigma$	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
$* q_2$	—	—

Transition table for DFA

$Q \setminus \Sigma$	0	1
$\rightarrow q_0$	q_0	q_0q_1
q_0q_1	q_0q_2	$q_0q_1q_2$
q_0q_2	q_0	q_0q_1
$q_0q_1q_2$	q_0q_2	$q_0q_1q_2$

<u>DFA</u>



Minimization of DFA

Task of changing a given DFA into equivalent DFA that has minimum no of states.

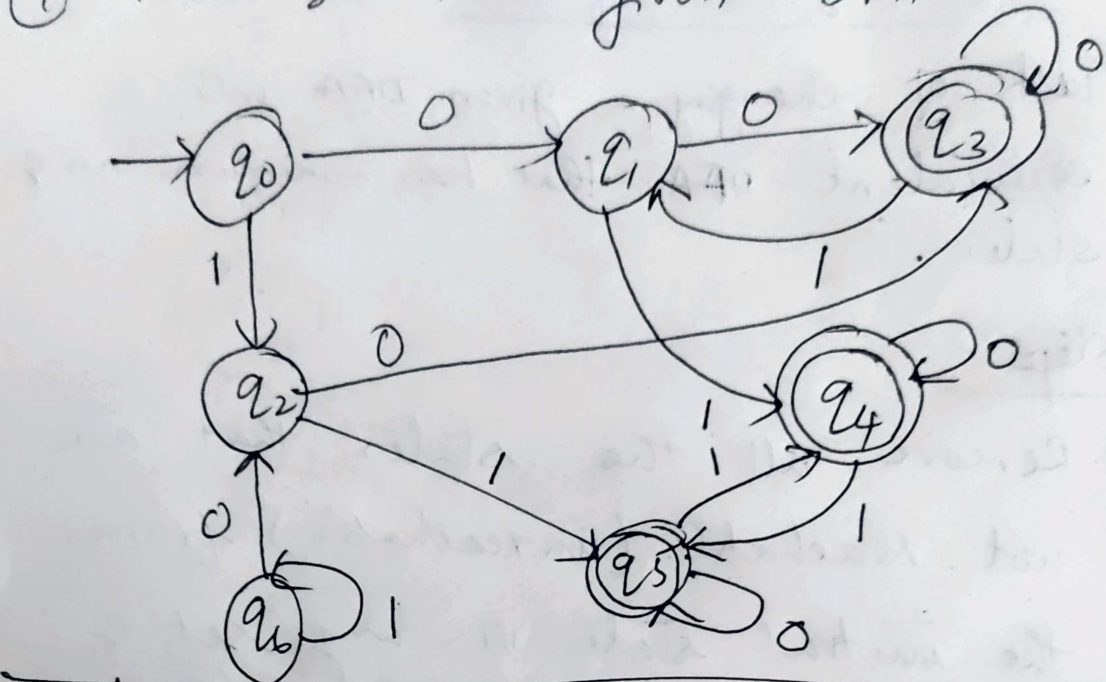
Steps:-

1. Remove all the states that are not reachable (unreachable) from the initial state via any set of transition of DFA

2. Split the transition table into 2 classes. Class 1 contains all final / accepting states and class 2 contains all non-final states.

3. After finding equivalence classes draw DFA combining the classes.

① Minimize the given DFA



Solution

Q \ Σ	0	1
→ q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
* q_3	q_3	q_1
* q_4	q_4	q_5
* q_5	q_5	q_4
q_6	q_2	q_6

Step 1:-

Remove the unreachable state

q_6 is unreachable

Step 2

$Q \backslash \Sigma$	0	1
$\rightarrow q_0$	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
$*q_3$	q_3	q_1
$*q_4$	q_4	q_5
$*q_5$	q_5	q_4

Step 2:-

Classes

① Final / accepting state
(q_3, q_4, q_5)

② Other states
$(q_0, q_1 q_2)$

Π_0 :- $\{q_0 \; q_1 \; q_2\}$ $\{q_3 \; q_4 \; q_5\}$

$\{q_0 q_1, 0\} = \underline{\underline{q_1 q_3}}$ belongs to different class

So not possible

$\{q_0 q_2, 0\} = \underline{q_1 q_3}$ belongs to different class

So not possible

$\{q_1 q_2, 0\} = q_3$
$\{q_1 q_2, 1\} = q_4, q_5$ belongs to same class

possible

So

$\Pi_1 = \{q_0\} \quad \{q_1 q_2\}$

$\{q_3\, q_4\, , q_5\}$

$\{q_3 q_4, 0\} = q_3, q_4$

$q_3 q_4, 1 \quad = q_1, q_5 \Rightarrow$ belongs to different class

Not possible

$\{q_3 q_5, 0\} = q_3\, q_5$

$\{q_3 q_5, 1\} = q_1, q_4 \Rightarrow$ belongs to different classes.

Not possible

$\left.\begin{array}{l} q_4 q_5, 0 = q_4, q_5 \\ q_4, q_5, 1 = q_4, q_5 \end{array}\right\}$ belongs to same class

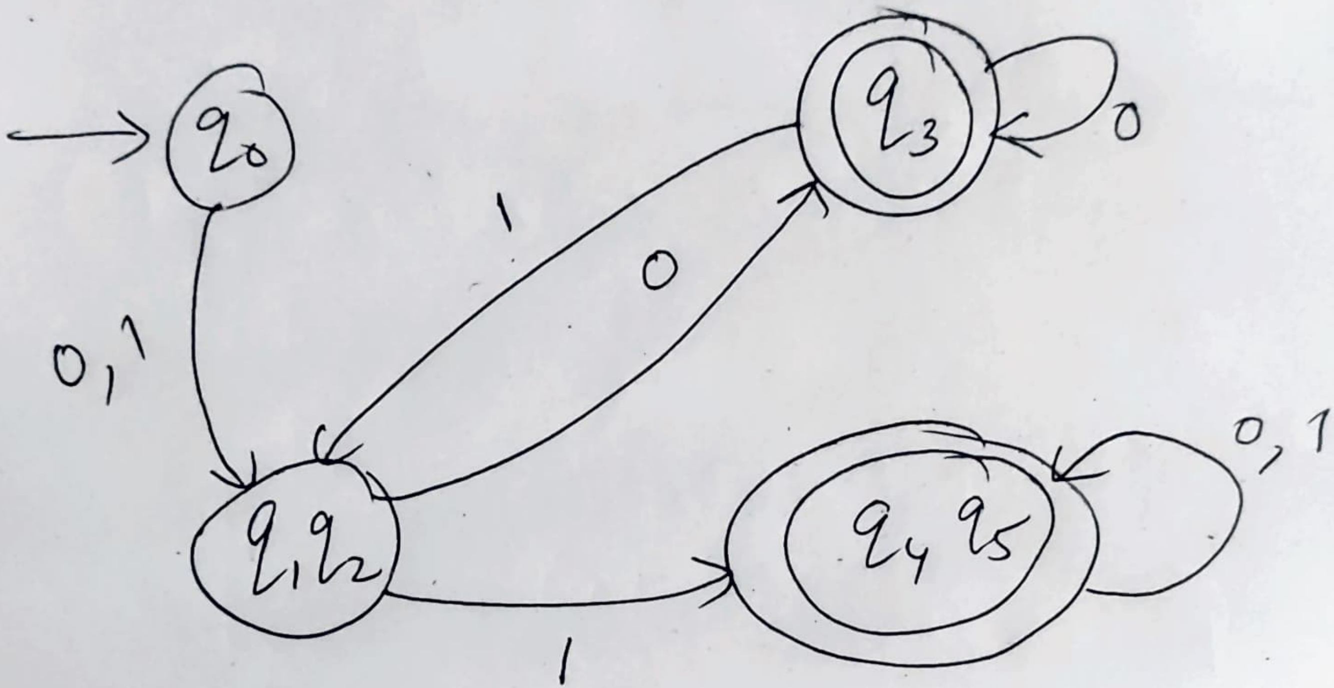
so

$\Pi_2 = \{q_3\} \quad \{q_4, q_5\}$

Equivalent classes are

$\{q_0\} \quad \{q_1 q_2\} \quad \{q_3\} \quad \{q_4 q_5\}$

Minimized DFA



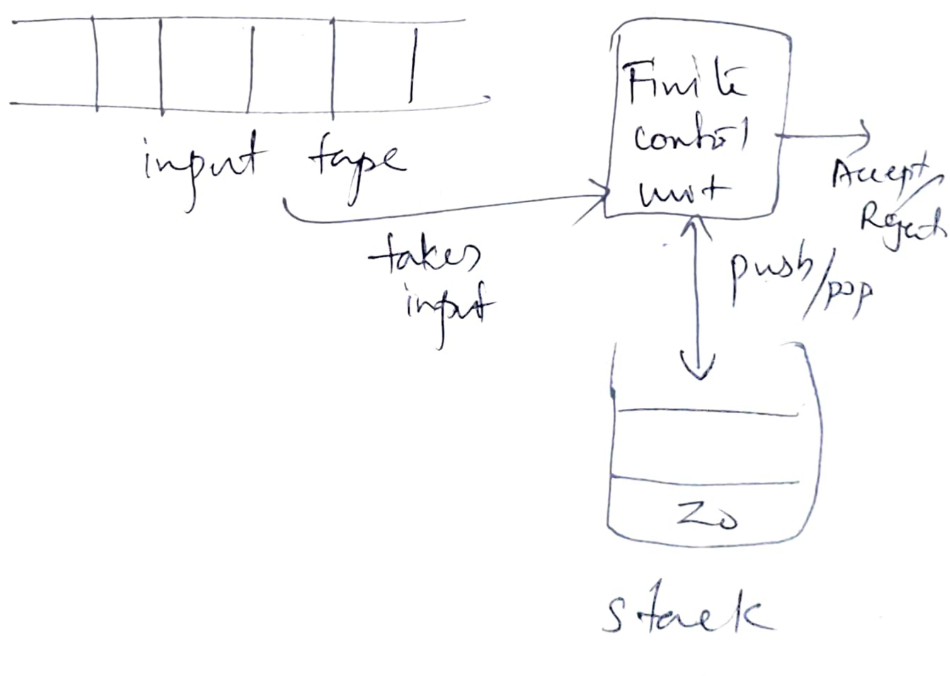
PushDown Automata (PDA)

→ is a way to implement a content free grammar (CFG) same way to design DFA for a regular grammar.

PDA has three components

- input tape
- control unit
- stack with infinite size

PDA has to read the top of the stack in every transaction.



Formal Definition

PDA described as 7-tuples

$$(Q, \Sigma, \ , \delta, q_0, Z_0, F)$$

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ input alphabet

$\Gamma \rightarrow$ Stack symbol

$\delta \rightarrow$ transition functions

$q_0 \rightarrow$ initial state

$Z_0 \rightarrow$ initial stack top symbol

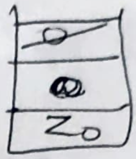
$F \rightarrow$ Final / Accepting state

① PDA to accept the following language by Final state

$$L = \{ 0^n 1^{2n} \mid n > 0 \}$$

strings :- { 011, 001111 ... }

Transition function δ



11

$$\delta(q_0, \underset{\text{input}}{0}, \underset{\text{stack top}}{Z_0}) = (q_0, 0Z_0)$$

001111

$$\delta(q_0, 0, 0) = (q_0, 00)$$

$$\delta(q_0, 1, 0) = (q_1, 0)$$

$$\delta(q_1, 1, 0) = (q_2, \varepsilon) \quad \text{*pop operation performed means } \varepsilon$$

$$\delta(q_2, 1, 0) = (q_1, 0)$$

$\delta(q_1, 1, 0) = (q_2, \varepsilon)$ <u>2nd 0 is popped</u>

$\delta(q_2, \varepsilon, Z_0) = \underline{(q_3, Z_0)}$ final state

<u>7-tuple</u>

$(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

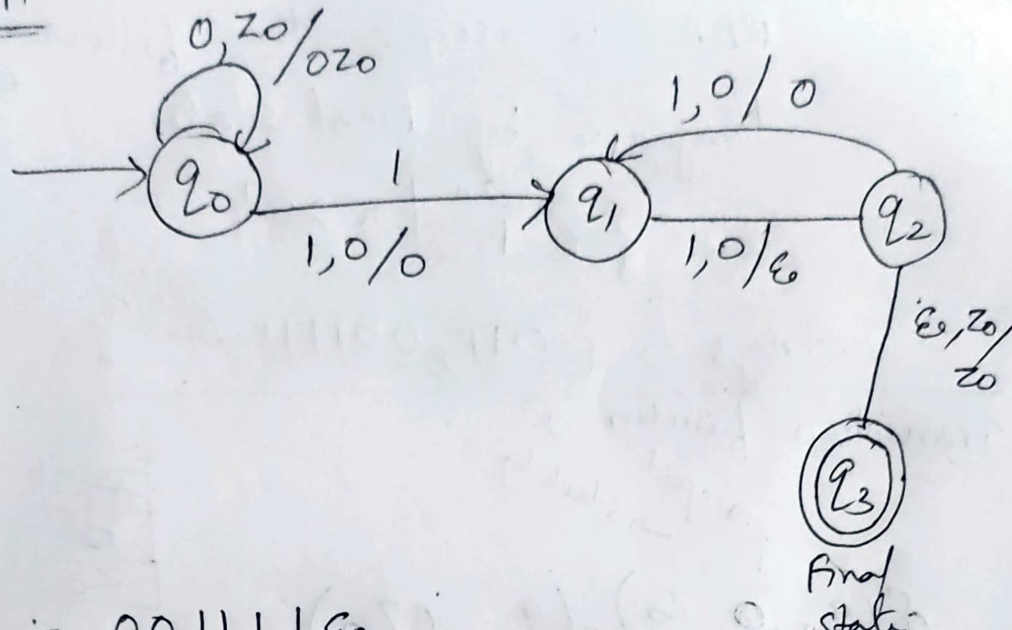
$Q = q_0, q_1, q_2, q_3$

$\Sigma = 0, 1$

$q_0 = q_0$

$F = q_3$

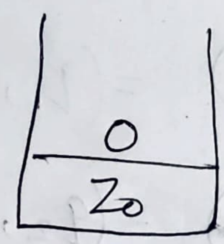
PDA



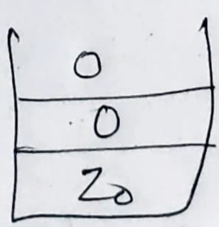
i/p :- 001111ε

Stack

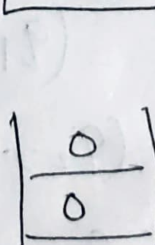
steps



① when 0 on i/p and Z0 on stack push 0



② when 0 on stack and 0 on i/p push 0

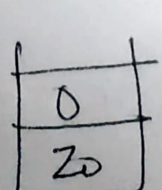


③ when 0 on stack and 1 on i/p do nothing move to new state, advance i/p

001

④ when 0 on stack and 2nd 1 on i/p, pop 0 and make ε

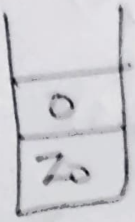
0011



(5) when 0 on stack i/p 1 go back to previous state

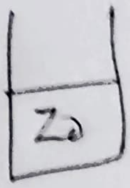
0 0 1 1 1 1 ε

(6) when 0 on stack and 2nd 1 on i/p pop 0



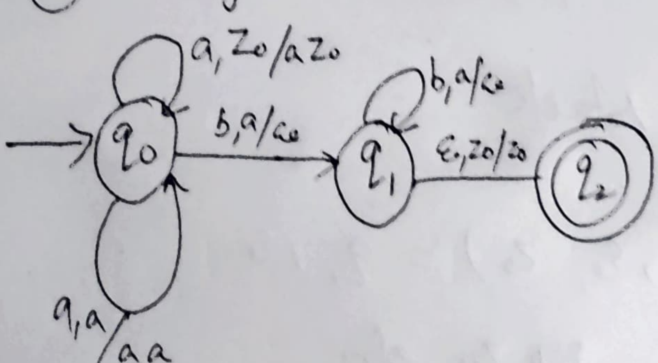
0 0 1 1 1 1 ε

(7) when i/p is ε and stack is Z0 final/accepting state

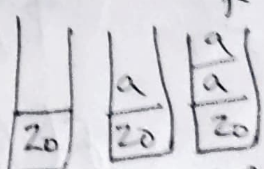


0 0 1 1 1 1 ε

(2) Design a PDA for $a^n b^n$ for $n > 0$



i/p:- a a b b ε
a a b b ε

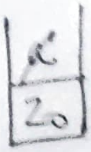


a, Z0 a, a

i/p:- a a b b ε

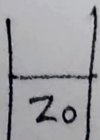
pop

i/p:- a a b b ε



pop

i/p:- a a b b ε



accept

$\delta(q_0, a, z_0) = q_0, az_0$ (push)

$\delta(q_0, a, a) = q_0, aa$ (push)

$\delta(q_0, b, a) = q_1, \varepsilon$ (pop)

$\delta(q_1, b, a) = q_1, \varepsilon$ (pop)

$\delta(q_1, \varepsilon, z_0) = q_2, z_0$

$\downarrow$

Final / Accepting state
state

Acceptance can be by Final state
or empty stack

If empty stack

$\delta(q_1, \varepsilon, z_0) = q_1, \varepsilon$

pop z_0 also

then acceptance by empty
stack