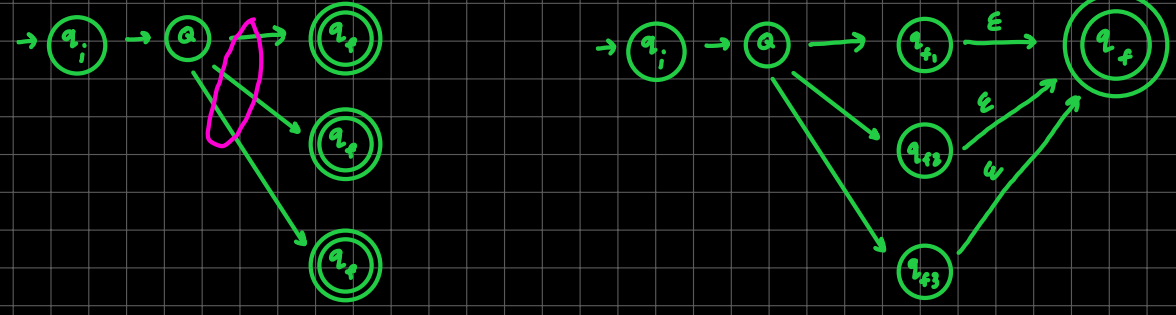
State Elimination

1- Initial state of DFA must not have any incoming edge,
if exists, create a new initial state

2- There must be only 1 final state present in DFA, convert all final states into non-final states and create a single final state



3- Final state of DFA must not have any outgoing edge
Create a new final state, having no outgoing edge from it

4- Eliminate all intermediary states one by one,
may be eliminated in any order

- At the end, only an initial state going to the final state will be left
- The cost of the transition will be the required R.E

- State elimination method can be applied to any finite automata
(NFA, ε-NFA, DFA ... etc)

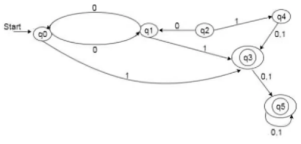
State Minimization

* Black-hole inclusion

- Equivalence partitioning algorithm to minimize DFA

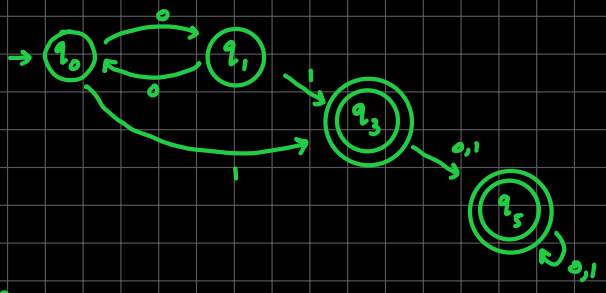
① Remove inaccessible states. After removing, repeat until no inaccessible states are present
② Seperate Final and non-final states
③ Check behaviour of each group/partition
④ Break states into groups of same behaviour, repeat until same behaviour is not obtained for every group.
⑤ Draw DFA, and extract R.E if needed by state elimination

Problem # 1: Minimize the following DFA and write the regular expression of the minimized machine.(10+10)

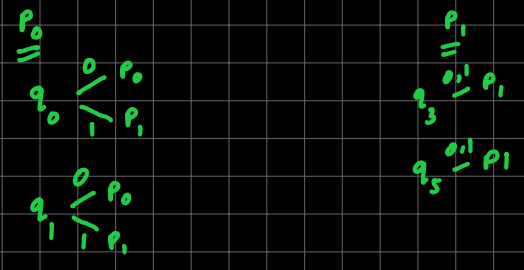


$q_0\ q_1\ q_3\ q_5$

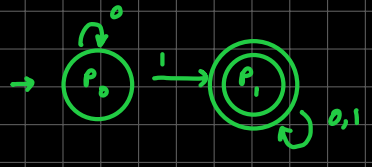
- inaccessible states : q_2, q_4



- Non-Final = $P_0 = \{q_0, q_1\}$
- Final = $P_1 = \{q_3, q_5\}$



- Same Behaviour, so either of them can't be broken further

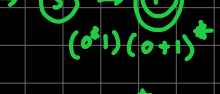


- Initial State = Partition in which initial state is present
- Final state= Partition in which final state is present

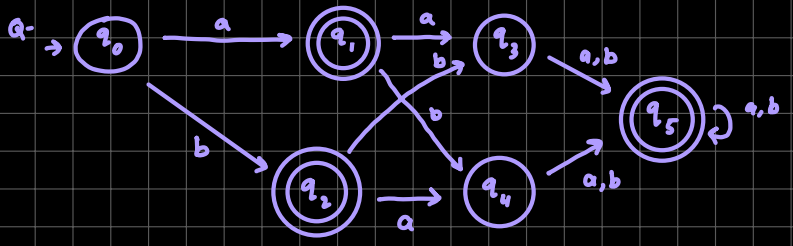
- Elimination

$(a+b)^* = a^* + b^*$

(step 1-3)



R.E = $0^*.1(0+1)^*$



- State Minimization
- No inaccessible state

$P_0 = \{q_0, q_3, q_4\}$ - Non-Final

$P_1 = \{q_1, q_2, q_5\}$ - Final

P_0

q_0 — a → P_1, b → P_1

q_3 — a → P_1, b → P_1

q_4 — a → P_1, b → P_1

P_1

q_1 — a → P_0, b → P_0

q_2 — a → P_0, b → P_0

q_5 — a → P_1, b → P_1

$P_2 = \{q_5\}$

q_5 — a → P_2, b → P_2

× —— ×

q_0 — a,b → P_1

q_3 — a,b → P_2

q_4 — a,b → P_2

$P_3 = \{q_3, q_4\}$

× —— ×

P_0

q_0 — a,b → P_1

P_2

q_5 — a,b → P_2

P_3

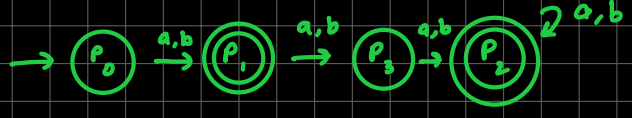
q_3 — a,b → P_2

q_4 — a,b → P_2

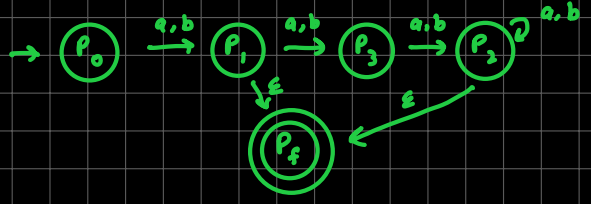
P_1

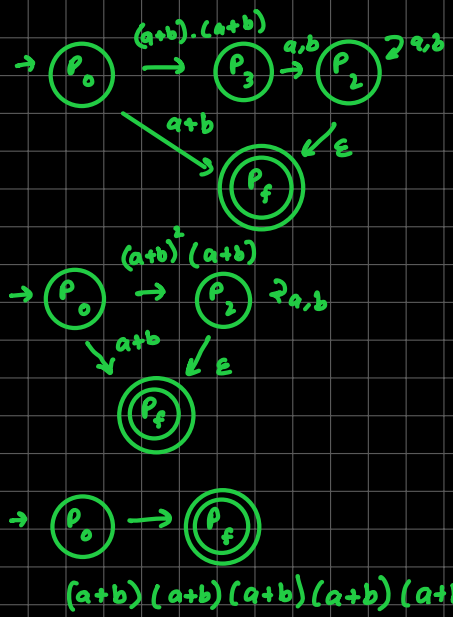
q_1 — a → P_3, b → P_3

q_2 — a → P_3, b → P_3



- State Elimination





Regular / Non-Regular Languages

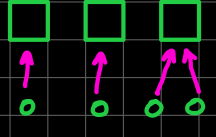
- Non-regular languages are those that can't be recognized by finite Automaton while regular languages can
- Pumping lemma as a tool to prove non-regular // non-context free languages
- Non-Regular:
- Infinite language
- Infinite state (if made through FA)
  ↳ PDA, CFG, TM

Regular	Non-Regular
· simple patterns · searching patterns · Network protocols · no memory beyond finite states	· Need memory · describe nested / balanced structures · Matching brackets · Programming lang syntax

Pumping lemma
(Negative test)

- If a language is regular, then long enough strings in it can be stretched (pumped) in the middle and still stay inside the language
- If pumping breaks the language, then the language is not regular

① Assume language L is regular
② If language is regular, then there must be a FA with n states such that FA→L, n states
③ Word should belong to L and |w| >= n
④ If ③ is true then there must be at least one state with loop
⑤ If ④ is true then w = xyz, where w = original string
x = Before y
y = Content of first loop of (w)
z = Remaining string



pigeons = Length of string
pigeonhole = states

⑥ Since $w \in L \; \forall \; w' \in L$

$w' = xy^i z \quad i \geq 0$

⑦ If any $w' \notin L$ then
Language is non-regular

Q- $RE = ac^*b$

FA:

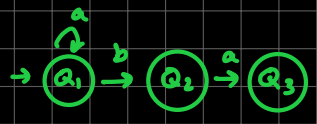
$w = acccb$

· Choose the word that causes revisiting of the state

$x = a$
$y = c$
$z = ccb$

$w = acb$
$x = a$
$y = c$
$z = b$

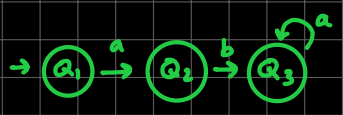
Case with X = null



$w = aba$

x = null
$Y = a$
$z = ba$

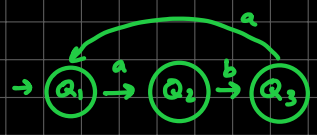
Case of Z null



$w = aba$

$X = ab$
$Y = a$
z = null

Case of X and Z null



$w = aba$

X = Null
Y = aba
z = Null

Range of Y = 1 to n
↳ can't be 0 as it would defy the pigeon-hole principle

Q- Prove language is regular or not using pumping lemma

$\Sigma = \{a, b\}$, $L = \{x \mid x \in \Sigma^* \text{ and } x = a^i b^j \text{ where } i \neq j \text{ and } 2i \neq j\}$

Let $w' = a^p b^{3p}$, where p is the length of pumping lemma, and there are p states

· Let w' become a regular language

x = null
$y = a^p$
$z = b^{3p}$

→ $xy^j z \; ; \; j \geq 0$

when $j = 3$, $(a^p)^3 b^{3p}$
$= a^{3p} b^{3p}$
$i \neq j$ condition violated,
hence not regular language

Grammar

- FA can only recognize regular languages
- Grammar gives theory, automata gives mechanism

- G = (N, T, S, P)

- G → Grammar
- N → Non-Terminals (Symbols that can be replaced) (Capital)
- T → Terminals (Actual Alphabet symbols of the language) (Not Capital)
- S → Start Symbol (where generation begins)
- P → Production rule (Rules for replacement)

a- S → aS | bS | a

a- S → aS | bA | ε
A → bA | ε

Production Rules

1. S → aS
2. S → bS
3. S → a

a(S)	aS
aa(S)	aaS
aab(S)	aabA
aaba	aabε
	aab

1- aab

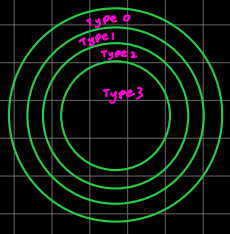
- L(G) = {aaba}

Types of Grammar

Type 0

- LHS → RHS

- LHS must have at least one non-terminal
  e.g : S → aS | bS | a



- Properties:
- No limits on how symbols can be replaced
- Can generate any language that a turing machine can recognize
- The most powerful class : Recursively Enumerable Language

Type 1 *

* Both rules satisfied but context not preserved, Type 1 or not!

- Context sensitive Grammar (CSG)

- Rules:
- Must be Type 0

- |LHS| ≤ |RHS|

- Properties:
- No ε-productions except S → ε if S doesn't appear on RHS
- Generated languages are context sensitive language
- Recognized by linear bounded automata (LBA)

e.g: aSb → aBb

(same) Left Context

(same) Right Context

if context preserved, then |LHS| ≤ |RHS|

Type 2

Type 3: Describes simple flat patterns, recognized by finite automata

- Context-Free Grammar
- CFG's describe non-regular languages

- Properties
- Must be Type 0
- Must be Type 1

Type 0: Most Powerful
↓ less powerful
Type 3: Least Powerful

- LHS must have exactly 1 Non-terminal

e.g: $S \to aS \mid bA \mid \varepsilon$
$A \to bA \mid \varepsilon$

Regular to Non-Regular Language

- Rules:
- Each production rule of Type 3 grammar must be linear and follow one of these patterns

→ Right-linear grammar — All non-terminals (if any) appear at the right end of RHS
$A \to aB$

→ Left-linear grammar — All non-terminals (if any) appear at the left end of RHS
$A \to Ba$

Note: grammar must be entirely right-linear or entirely left-linear

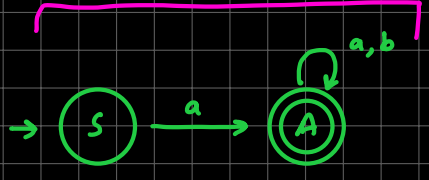
* If mixed, then chances are of non-regular grammar

Cases of Regular Grammar

1. Start with a

- $a(a+b)^*$

NFA



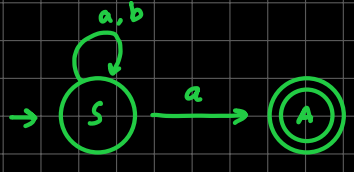
$(a+b)^*$
$S \to aS \mid bS \mid \varepsilon$

- Regular Grammar

$S \to aB$
$B \to aB \mid bB \mid \varepsilon$

2. End with a * Ghalat lg rhi hai

$(a+b)^*a$

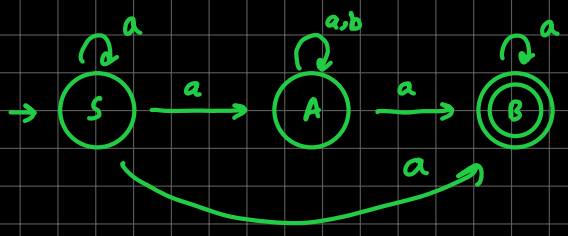


Grammar:

$S \to aS \mid aS \mid aA$
$A \to aA \mid \varepsilon$

3. Start with a
End with a

$a(a+b)^*a$

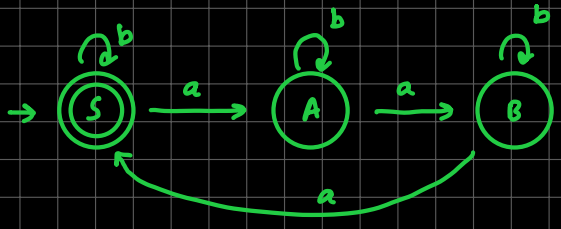


Grammar

$S \rightarrow aA$
$A \rightarrow aA \mid bA \mid aB$
$B \rightarrow aB \mid \varepsilon$

4- Divisible

- No of a's divisible by 3



- Grammar
$S \rightarrow aA \mid bS \mid \varepsilon$
$A \rightarrow aB \mid bA$
$B \rightarrow aS \mid bB$

5. Substrings:
have sub-string ba

- Grammar
$S \rightarrow aS \mid bS \mid bA$
$A \rightarrow aB$
$B \rightarrow aB \mid bB \mid \varepsilon$

6. At-least
At most
Exactly 1 or more

At least: $(a+b)^*a(a+b)^*$
At most: $b^*(a+\varepsilon)b^*$
Exactly 1: b^*ab^*

- Type 2
  | bridge linear grammar
- Type 3

7- Finite language

- Simple NFA

8- No of a's followed by No. of b's

→ ordered
$a^n b^m \mid n\%3 = 0$

→ Unordered
$n_a(w), n_b(w) \%3 = 2$

Non-Regular Language

- Infinite states
- Infinite patterns

- $a^n b^n \mid n \geq 0$

$S \to aSb \mid \varepsilon$

Cases

Addition

- $a^n b^n \mid n \geq 0$
- $S \to aSb \mid \varepsilon$

$a^{n+1} b^n \mid n \geq 0$ when $n = 0$, a

$S \to aSb \mid a$

Find Base Case

$a^{n+1} b^{n+2} \mid n \geq 0$ when $n = 0$, ab^2

$S \to aSb \mid abb$

Multiplication

$a^n b^{2n} \mid n \geq 0$, when $n = 0$, null

$S \to aSbb \mid \varepsilon$

- $a^{2n+3} b^{3n+2} \mid n \geq 0$, $n = 0$, $a^3 b^2$

$S \to aaSbbb \mid aaabb$

- $a^n b^m \mid n = m - 1$, $m \geq 1$
- $a^{n-1} b^n \mid n \geq 1$, $n = 1$, b

$S \to aSb \mid b$

Greater Than or equal to

- $a^n b^m \mid n > m$

$S \to aSb \mid aA$
$A \to \varepsilon \mid aA$

- $a^n b^m \mid n \geq m$

- $S \to aSb \mid aA \mid \varepsilon$
- $A \to \varepsilon \mid aA$

Less than or equal to

- $a^n b^m \mid n < m$

$S \to aSb \mid bA$
$A \to \varepsilon \mid bA$

- $a^n b^m \mid n \leq m$

- $S \to aSb \mid bA \mid \varepsilon$
- $A \to \varepsilon \mid bA$

Not Equal to

- $a^n b^m \mid n \neq m$

$n < m$, $n > m$

- $S \to A \mid B$
  $S \to aAb \mid aA \mid a$
  $B \to aBb \mid bB \mid b$

Linear Growth

- $a^m b^n \mid m \geq 2n$ $n = 1, m \geq 2$

- $S \to aaSb \mid \varepsilon \mid aS$

Range Dependent

- $a^m b^n \mid m \leq 2n$

$n = 3$, $m \leq 6$
$m = [0, 6]$

- $S \to \varepsilon \mid Sb \mid aSb \mid aaSb$

- A CFG can only binary (pairwise) comparisons - it can ensure equality b/w two groups of symbols at a time like $a^n b^m c^n$

- $L = (a^n b^m c^n)$ $n, m \geq 0$ $\quad L2 = \{ a^n b^m c^n d^n \mid n, m \geq 0 \}$

- $S \rightarrow aSc \mid A \mid \varepsilon$
- $A \rightarrow bAc \mid \varepsilon$

- CSG multi-stack required

Stack example
push and pop after comparison

- Unordered
- $L = (na(w) = nb(w))$

$S \rightarrow aSb \mid bSa \mid \varepsilon \mid SS$

- Palindrome:

- Even= abba | baab $\quad$ odd= abab | babab

$S \rightarrow aSa \mid bSb \mid \varepsilon$ $\quad$ $S \rightarrow aSa \mid bSb \mid a \mid b$

- Closure Properties: (CFL)

① Union

$L_1 = \{ a^n b^n \mid n \geq 0 \}$ $\quad L_1 \cup L_2$

$L_2 = \{ c^m d^m \mid m \geq 0 \}$

- $S \rightarrow S1 \mid S2$
  $S1 \rightarrow aS1b \mid \varepsilon$
  $S2 \rightarrow cS2d \mid \varepsilon$

② Concatenation: $L_1 \cdot L_2$

$L_1 = \{ a^n b^n \mid n \geq 0 \}$

$L_2 = \{ c^m d^m \mid m \geq 0 \}$

$L_1 \cdot L_2 = \{ a^n b^n c^m d^m \mid n, m \geq 0 \}$

- $S \rightarrow S1.S2$
  $S1 \rightarrow aS1b \mid \varepsilon$
  $S2 \rightarrow cS2d \mid \varepsilon$

③ Transpose

$L_1 = \{ a^n b^n \mid n \geq 0 \}$

$L_2 = \{ c^m d^m \mid m \geq 0 \}$

$L_2 \cdot L_1 = \{ c^m d^m a^n b^n \mid m, n \geq 0$

- $S \rightarrow S2\,S1$
  $S1 \rightarrow aS1b \mid \varepsilon$
  $S2 \rightarrow cS2d \mid \varepsilon$

For single language transpose flip the var positions

- Not closed Properties: (CSL) can't make grammar

- Intersection

- $L_1 = \{ a^m b^m c^n \mid m, n \geq 0 \}$ $\quad a^m, b^m$ compared but depend on how many c^n present

$L_2 = \{ a^n b^m c^m \mid m, n \geq 0 \}$ $\quad b^m, c^m$ compared but depend on how many c^m present

- Set Diff
- $A-B = A \cap B^c$

- Demorgan
- $(A \cup B)^c = A^c \cap B^c$

intersection present in both so not closed

PDA

- PDA = FA + one infinite stack

- 7-tuple Model

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

Q = States
Σ = input symbol
δ = transition function
q_0 = initial state

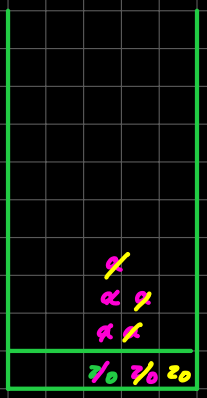
Z_0 = Initial Stack Symbol
F = Final State
Γ = Stack Symbols

Q- $a^n b^n \mid n \geq 1$

a a a b b b

push

1. $\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$ - Initial
pop
2. $\delta(q_0, a, a) \rightarrow (q_0, aa)$ - For a
3. $\delta(q_0, b, a) \rightarrow (q_1, \varepsilon)$ - For b
4. $\delta(q_1, \varepsilon, Z_0) \rightarrow (q_f, Z_0)$ - Final



Acceptance Criteria:

1. Stack is empty
2. Final State

PDA

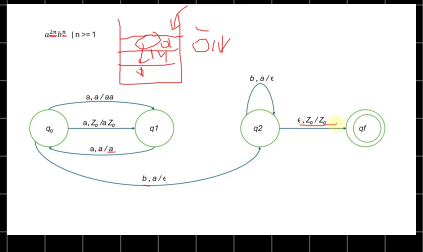
NPDA
if any state have both true and null move

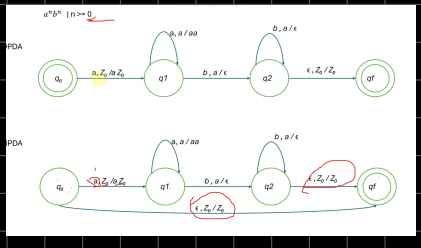
DPDA
If every state have either true move or null move but not both

- True Move
$\delta: (q_0, a, Z_0) \rightarrow (q_1, AZ)$

- Null Move:
$\delta: (q_0, \varepsilon, Z_0) \rightarrow (q_1, AZ)$

Power
NPDA > DPDA





Turing Machines

- A mathematical model of computation that describes an abstract machine capable of simulating any computer algorithm

- An abstract machine with:

1. An infinite tape
2. A read/write head
3. Finite set of states
4. A transition function → (read symbol → write symbol → move left/right → change state)

- TM can act as an acceptor (FA, PDA, LBA) or as a transducer (Mealy, Moore)

- 7-tuple model
- $M = (Q, \Sigma, \Gamma, \delta, \text{blank}, q_0, q_f)$

- Q = set of states
- Σ = Input Alphabet
- Γ = Tape Alphabet
- δ = Transition Function
- blank = The blank symbol

$a^{2n}b^{n}$ | n >= 1