Eg: $a^{n^2} \mid n \geq 1$ $\longrightarrow$ Not Regular

$L = \{a, a^4, a^9, a^{16} \dots\}$

not in AP

Eg: $a^{2^n} \mid n \geq 1$ $\longrightarrow$ Not Regular

$L = \{a^2, a^4, a^8, a^{16} \dots\}$

Eg: $a^i b^{j^2} \mid i, j \geq 1$ $\longrightarrow$ Not Regular

$a^i = \{a, aa, aaa \dots\}$

$= \underbrace{\{a^1, a^2, a^3 \dots}_{\text{FA ✓}}\}$

$b^{j^2} = \underbrace{\{b, b^4, b^9, \dots\}}_{\text{FA ✗}}$

Eg: $a^i b^{2^n}$ Not Regular

a^i → FA ✓ b^{2^n} → FA ✗

Eg: $a^i b^p \mid i \geq 1$, p is prime

a^i → FA ✓ b^p → FA ✗

Not Regular

Eg: $w \mid n_a(w) = n_b(w)$ $\Sigma = \{a, b\}$

1 2 3

a b a a b b

1 2 3

Not Regular

a's b's

store a's

store b's

Eg: $w \mid n_a(w) \leq n_b(w)$

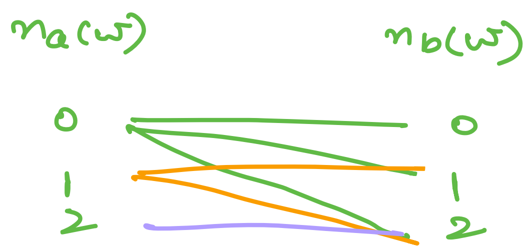
Eg: $w \mid n_a(w) \geq n_b(w)$

FA doesnot provide storage hence you can't keep track of a's & b's.

Eg: $n_a(w) \bmod 3 \leq n_b(w) \bmod 3$

$3 \times 3 = 9$ states

FA can do modular counting

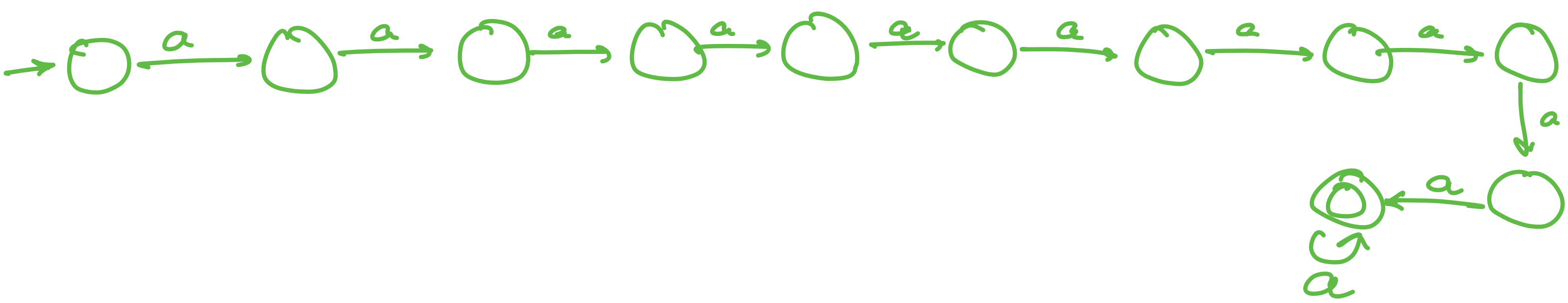


Regular ✓

Eg: $a^n \mid n \geq 10$ Regular

$L = \{a^{10}, a^{11}, a^{12} \dots\}$

Strings of length atleast 10



Eg: $w w w^R \mid w \in (a,b)^*$ Not Regular

Store w

Compare w

Compare in rev. order

Eg: $a^n b^{n+m} c^m \mid n, m \geq 1$ Not Regular

$\underbrace{\qquad}_{a's}^{n} \underbrace{\qquad}_{b's}^{n} \underbrace{\qquad}_{b's}^{m} \underbrace{\qquad}_{c's}^{m}$

Eg: $wxw^R \mid w, x \in (0,1)^+$

$w = 110$
$x = 101$

$wxw^R = 110\,\underbrace{\underline{101}_{x}\,011}_{x}$

$1 \mid \underbrace{10\ 101\ 01}_{x} \mid 1$ w? x?

wwR is not regular

$$\underbrace{1}_{w}\ \underbrace{10\ 101\ 01}_{x}\ \underbrace{1}_{w^R}$$

RE : $0(0+1)^*0 + 1(0+1)^*1$ Regular language

Eg: $wxw^R \mid w \in (0,1)^+$ $\underline{|x| = 5}$

↳ you can't extend x beyond this range

Not Regular

Closure Properties of Regular languages

RL are closed under union, intersection, concatenation, complementation & kleene closure

Closed operation:

Set of all natural nos

N

Natural nos are closed under addition

Closed ??

2 natural nos

$2 + 3 = 5$

Result is also a natural no.

RL

$\cdot R_1$

$\cdot R_2$

$R_1 \text{ op } R_2 = \underline{\underline{R_3}}$

if R3 also belongs to RL then u will say RL are closed under op operation

Union:

L_1 & L_2 are Regular languages
$L_1 \cup L_2$ will also be Regular

If L_1 is regular then R_1 is RE corresponding to it

If L_2 ———— R_2 ————

$L_1 \cup L_2$

$\downarrow \quad \downarrow$

$R_1 \quad R_2$

$L_1 \cup L_2 \longrightarrow R_1 + R_2$

R1: $(a+b)$ R2: $(a \cdot b)$

$\underbrace{(a+b) + (a \cdot b)}_{RE.}$

Concatenation:

$L_1 \quad L_2$ are RL

$\downarrow \quad \downarrow$

RE: $R_1 \quad R_2$

$L_1 . L_2 \longrightarrow R_1 . R_2$

↳ Regular language

Kleene Closure

L_1 is a RL
↓
R_1 is RE

RE: (a+b)
$(a+b)^*$
RE
RL.

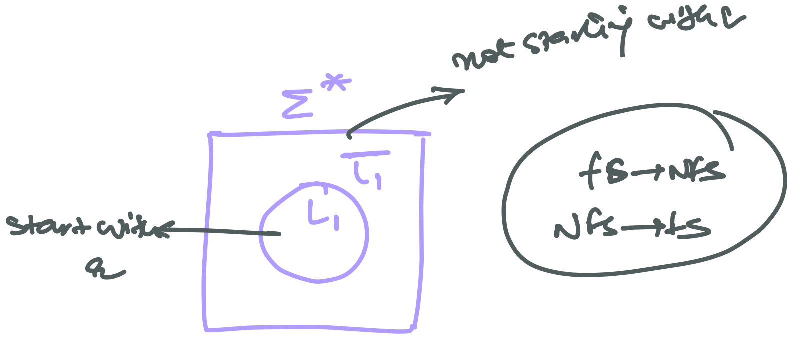
$L_1^* \longrightarrow R_1^* \longrightarrow R_1^*$ is also Regular.

Complementation:

L_1 is a RL

$\overline{L_1} = \Sigma^* - L_1$

↳ Complement of L_1 is also Regular.



L_1 is a RL $\longrightarrow$ DFA $\longrightarrow (Q, \Sigma, \delta, q_0, F)$

$\longrightarrow \overline{DFA} \longrightarrow (Q, \Sigma, \delta, q_0, Q-F)$

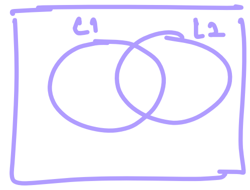
$\overline{L_1}$

Intersection:

$L_1 \cap L_2$

↓ RL ↓ RL

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



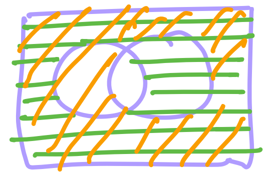
$\overline{L_1} \longrightarrow$ Regular

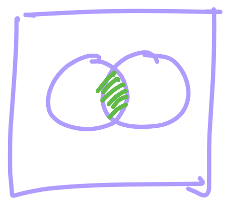
$\overline{L_2} \longrightarrow$ Regular

$\overline{L_1} \cup \overline{L_2} \longrightarrow$ Regular

$\overline{\overline{L_1} \cup \overline{L_2}} \longrightarrow$ Regular

$L_1 \cap L_2 \longrightarrow$ Regular





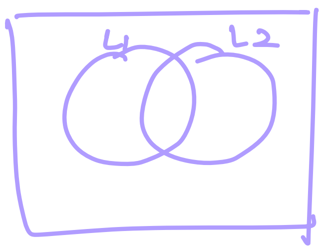
Difference

RL are closed under difference.

$L_1 - L_2$ will also be Regular

L_1 ↓ Regular, L_2 ↓ Regular

$L_1 - L_2 = L_1 \cap \overline{L_2}$

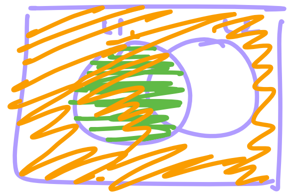


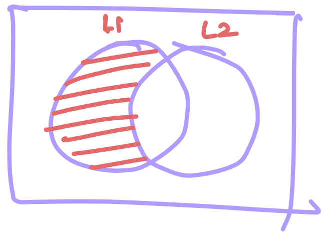
$L_1 \longrightarrow$ RL

$L_2 \longrightarrow$ RL

$\overline{L_2} \longrightarrow$ RL

$L_1 \cap \overline{L_2} \longrightarrow$ RL





Reversal

L $\rightarrow$ RL

$L^R \rightarrow$ RL   To prove

L $\longrightarrow$ Regular $\longrightarrow$ DFA

$\downarrow$

Reverse DFA

$\downarrow$

$L^R \longleftarrow$ FA

Reverse DFA

$\downarrow$

exchanging IS & FS
reverse dir^n of edges

$$\underset{(L)}{\text{DFA}} \xrightarrow{\text{Reversal}} \underset{(L^R)}{\text{FA}} \checkmark$$

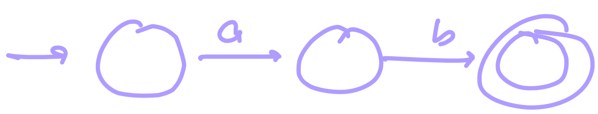
Regular languages are not closed under Infinite Union:

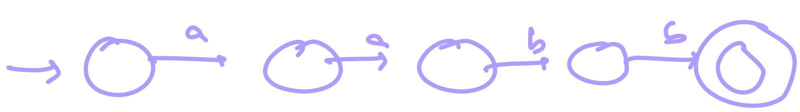
↳ If u do union of RL infinite times result may not be regular.

$L_1 = \{a^1 b^1\}$

$L_2 = \{a^2 b^2\} \rightarrow$ Regular

$L_3 = \{a^3 b^3\}$





$L_1 \cup L_2 \cup L_3 \cdots\cdots = \{a^n b^n \mid n \geq 1\}$

not Regular

Decidability Property of Regular Languages:

Decidability: Algo → terminate

→ **Emptiness problem is decidable:**

↳ FA is not accepting any string

When can u say that FA will accept at least one string?

→ If ur FA is having at least 1 FS & that FS should be reachable from IS.

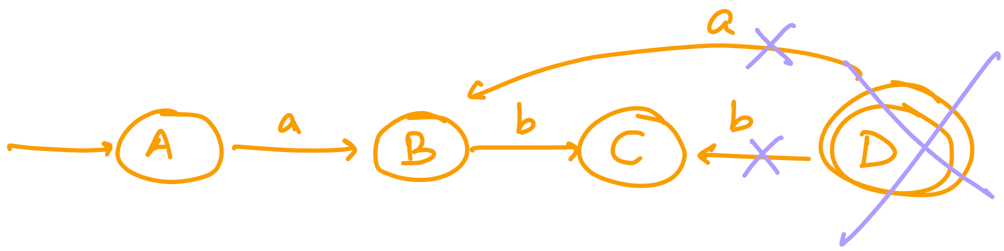
Algo:

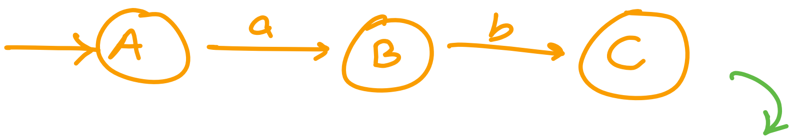
1. Select all states which are not reachable from IS. Delete all unreachable states & also delete transitions corresponding to them.

2. In the remaining FA, see if there is atleast 1 final state

True: FA will accept at least 1 string.

false: FA won't accept any string.





FA won't accept any string
(language accepted by FA is empty)

→ Infiniteness problem is Decidable:

Finite language:

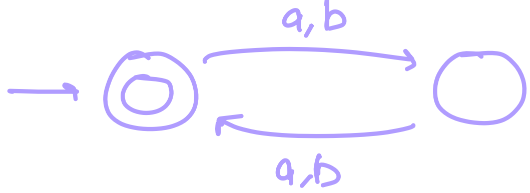
$\Sigma: \{a, b\}$

L_1 = Strings of length 2
= {aa, ab, ba, bb} → finite

L_2 = strings of even length
= { ε, aa, ab, aaaa, abba .... }
↳ infinite

Infiniteness:
↳ Given a language, we will be able to tell if it is finite or infinite

Algo:



strings of even length

ε, ab, abba

If a language is infinite, then definitely ur DFA will have a loop.

1. loop
2. reachable from IS
3. from loop reach to fs.

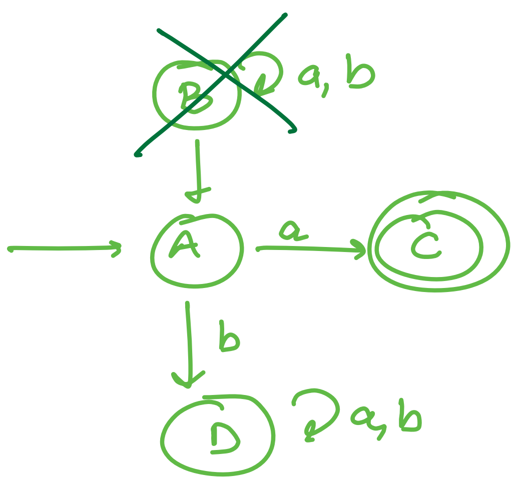
Algo:

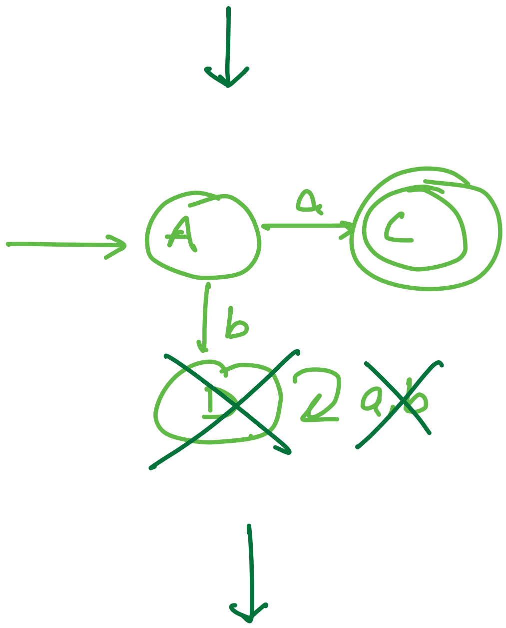
1. Remove all states which are not reachable from IS. and also transitions corresponding to them.

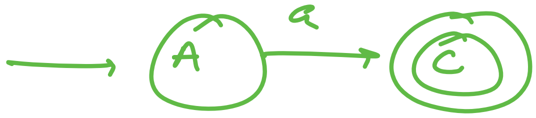
2. delete the states & transitions from which u can't reach to final state.

3. In remaining FA, if there is atleast 1 loop → True: infinite

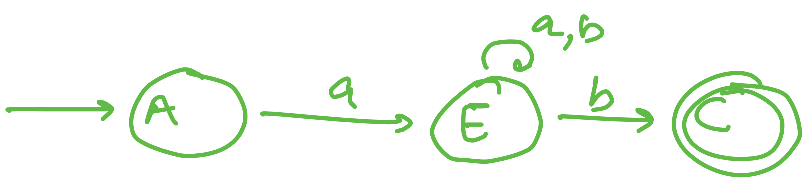
   False: finite







no loop is present : finite



loop is present : Infinite

Equality Problem is Decidable :

$L_1 = L_2$?

If strings generated by these 2 languages are same then $L_1 = L_2$

Algo:

RL :	L_1	L_2
	$\downarrow$	$\downarrow$
DFA :	DFA_1	DFA_2
	$\downarrow$	$\downarrow$
minimized DFA :	DFA m1	DFA m2

compare minimized DFA 1 & minimized DFA 2

$\rightarrow$ Equal

True $\rightarrow$ $L_1 = L_2$

False $\rightarrow$ $L_1 \neq L_2$

Membership problem is Decidable:

$\rightarrow$ Language L, string w

$w \in L$?

Algo: RL L $\longrightarrow$ FA

To FA provide w, if at end final state

True $\rightarrow$ Accepted

false $\rightarrow$ Not Accepted

Regular Expression

Representation

Regular Language

Generates → Regular Grammar

Acceptor → Finite Automata

Grammar:

Intention behind Grammar is to generate the entire language.

Grammar is represented by:

(V, T, P, S)

V → Variables

T → Terminal

P → Production

S → Start Symbol

Eg:

Start Symbol

Production P

$$S \rightarrow aSB$$
$$S \rightarrow aB$$
$$B \rightarrow b$$

$V = \{S, B\}$

$T = \{a, b\}$

Eg:

Start Symbol

Rules

$$S \rightarrow aSB$$
$$S \rightarrow aB$$
$$B \rightarrow b$$

$V = \{S, B\}$

$T = \{a, b\}$

Language generated by this grammar?

$S \rightarrow aSB$

$S \rightarrow aaSBB$

$S \rightarrow aaaBBB$

$S \rightarrow aaabBB$

$S \rightarrow aaabbB$

$S \rightarrow aaabbb$

$a^n b^n$

Derivation: Deriving a string from the grammar starting from start symbol.

Derivation of aabb:

Sentential form / sequential form

$S \rightarrow aSB$

$\rightarrow aaBB$

$\rightarrow aabB$

$\rightarrow aabb$

left symbol = left most derivation

right symbol: right most derivation

Derivation Tree / Parse Tree:

aabb