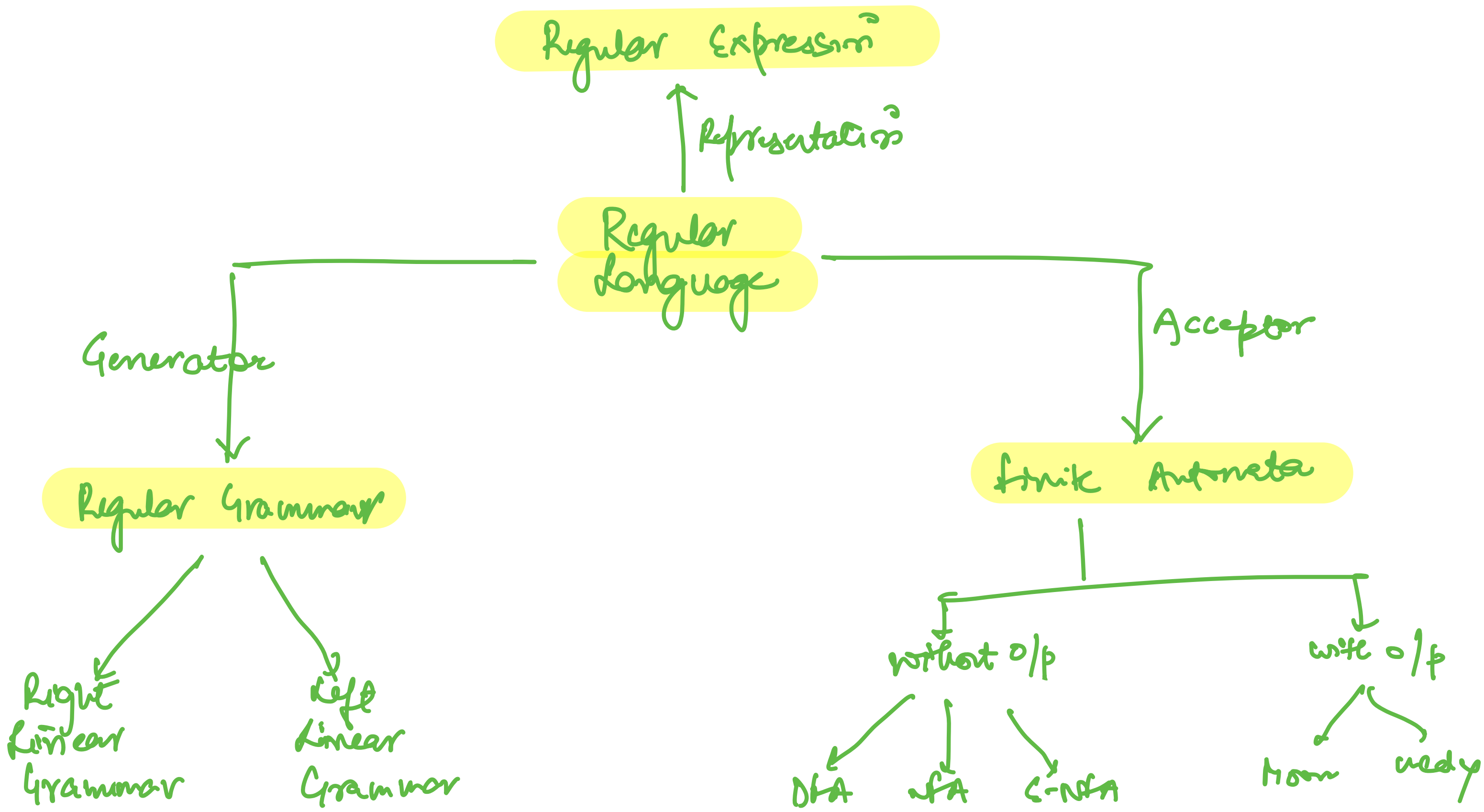


Regular Expression

Regular Expression are the representation of languages which are accepted by FA.

3 operators:

i) + (union) $\quad a+b \rightarrow$ either a or b

ii) · (concatenation) $\quad a \cdot b \rightarrow a \cdot b$

iii) * (Kleene closure)

a) Primitive RE

$\phi, \varepsilon, \Sigma$ (input alphabet)

b) r_1, r_2 are RE

$r_1 + r_2$, $r_1 \cdot r_2$, r_1^* are RE

c) can apply a and b as many times as you want

$((a+b)a\cdot b)^*$

$\phi = \{\ \}$

$\epsilon = \{\epsilon\}$

$a = \{a\}$

$a^* = \{\epsilon, a, aa, aaa \ldots\}$ $\quad * \to 0, 1, 2, 3 \ldots$

$a^+ = \{a, aa, aaa \ldots\}$ $\quad + \to 1, 2, 3 \ldots$

$a \cdot a^*$

$(a+b)^*$: string using a, b $\quad (a+b).(a+b).(a+b).(a+b)$ → a b a a

$(a+b)^3$: $\underbrace{(a+b)(a+b)(a+b)}_{\text{3 length string}}$ → a b a | strings of length 3

Algebraic properties of regular expressions:

Kleene closure is an unary operator and Union(+) and concatenation operator(.) are binary operators.

1. Closure:

(a+b) (a.b)

If r1 and r2 are regular expressions(RE), then

r1* is a RE
r1+r2 is a RE
r1.r2 is a RE

2. Closure laws –

(r*)* = r*, closing an expression that is already closed does not change the language.
∅* = ∈, a string formed by concatenating any number of copies of an empty string is empty itself.
r+ = r.r* = r*r, as r* = ∈ + r + rr+ rrr …. and r.r* = r+ rr + rrr ……
r* = r*+ ∈

3. Associativity –

If r1, r2, r3 are RE, then

i.) r1+ (r2+r3) = (r1+r2) +r3
For example : r1 = a , r2 = b , r3 = c, then

The resultant regular expression in LHS becomes a+(b+ c) and the regular set for the corresponding RE is {a, b, c}.
for the RE in RHS becomes (a+ b) + c and the regular set for this RE is {a, b, c}, which is same in both cases. Therefore, the associativity property holds for union operator.

ii.) r1.(r2.r3) = (r1.r2).r3
For example – r1 = a , r2 = b , r3 = c
Then the string accepted by RE a.(b.c) is only abc.
The string accepted by RE in RHS is (a.b).c is only abc ,which is same in both cases. Therefore, the associativity property holds for concatenation operator.
Associativity property does not hold for Kleene closure(*) because it is unary operator.

4. Identity –
In the case of union operators
if r+ x = r ⇒ x= ∅ as r ∪ ∅= r, therefore ∅ is the identity for +.

Therefore, ∅ is the identity element for a union operator.

In the case of concatenation operator –
if r.x = r , for x= ∈
r.∈ = r ⇒ ∈ is the identity element for concatenation operator(.) .

5. Annihilator –
If r+ x = x ⇒ r ∪ x= x , there is no annihilator for +

In the case of a concatenation operator, r.x = x, when x = ∅, then r.∅ = ∅, therefore ∅ is the annihilator for the (.)operator. For example {a, aa, ab}.{ } = { }

6. Commutative property –
If r1, r2 are RE, then

r1+r2 = r2+r1. For example, for r1 =a and r2 =b, then RE a+ b and b+ a are equal. a+b=b+a
r1.r2 ≠ r2.r1. For example, for r1 = a and r2 = b, then RE a.b is not equal to b.a. a.b ≠ b.a

7. Distributed property –
If r1, r2, r3 are regular expressions, then

(r1+r2).r3 = r1.r3 + r2.r3 i.e. Right distribution
r1.(r2+ r3) = r1.r2 + r1.r3 i.e. left distribution
(r1.r2) +r3 ≠ (r1+r3)(r2+r3)

8. Idempotent law –
r1 + r1 = r1 ⇒ r1 ∪ r1 = r1 , therefore the union operator satisfies idempotent property.

r.r ≠ r ⇒ concatenation operator does not satisfy idempotent property.

9. Identities for regular expression –
There are many identities for the regular expression. Let p, q and r are regular expressions.

∅ + r = r
∅.r= r.∅ = ∅
∈.r = r.∈ =r

$\epsilon^* = \epsilon$ and $\varnothing^* = \epsilon$

$r + r = r$

$r^*.r^* = r^*$

$r.r^* = r^*.r = r+.$

$(r^*)^* = r^*$

$\epsilon + r.r^* = r^* = \epsilon + r.r^*$

$(p.q)^*.p = p.(q.p)^*$

$(p + q)^* = (p^*.q^*)^* = (p^* + q^*)^*$

$(p+ q).r= p.r+ q.r$ and $r.(p+q) = r.p + r.q$

Reference Link : https://www.geeksforgeeks.org/properties-of-regular-expressions/

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

RE for strings of length exactly 2

$L = \{aa, ab, ba, bb\}$ → finite apply union b/w all these strings

$aa + ab + ba + bb$

$a(a+b) + b(a+b)$

$\underbrace{(a+b)}_{\text{either a or b}}.\underbrace{(a+b)}_{\text{either a or b}}$

Q: lengths exactly 3

$(a+b)(a+b)(a+b)$

Q: lengths atleast 2 $\Sigma = \{a,b\}$

$L = \{aa, ab, ba, bb, aaa, aab \dots \}$

— — ⊖ ?

$\underbrace{(a+b)(a+b)}_{\text{length is exactly 2}}\underbrace{(a+b)^*}_{\text{more than 2 lengths}}$

* replaced by 2 : length : 4

* ——— 3 : length 5

⋮

Q: length atmost 2

0 length + 1 length + 2 lengths

$L = \{\varepsilon, a, b, aa, ab, ba, bb\}$

1 way: $\varepsilon + a + b + aa + ab + ba + bb$

2 way: $(\varepsilon + a + b)\ (\underline{\varepsilon + a + b})$

length 1: ε $\quad a \quad = \varepsilon . a = a$

length 0: ε $\quad \varepsilon \quad = \varepsilon . \varepsilon = \varepsilon$

length 2: a $\quad b \quad = a . b$

Q: Even length String $\quad \Sigma = \{a, b\}$

$L = \{\varepsilon, aa, ab, ba, bb, aaaa, \ldots\}$

0 2 4 6 8 ....

$*:0$: 0 length : ε
$*:1$: 2 length
$*:2$: 4 length
$*:3$: 6 length
⋮

$((a+b)(a+b))^*$: Repeat pair of 2's

$((a+b)(a+b))^2$

$(a+b)(a+b)\ (a+b)(a+b)$

Q: odd length String

length 1, 3, 5, 7, 9 ....

$\underbrace{\underbrace{((a+b)(a+b))^*}_{\text{Even}}\ (a+b)}_{\text{odd}}$ | $\underbrace{(a+b)\ \underbrace{((a+b)(a+b))^*}_{\text{Even}}}_{\text{odd}}$

Q: String whose length is $\cong 2 \bmod 3$

↳ on dividing the string length by 3, remainder will be 2.

$$(a+b)^{3n+2} \mid n \geq 0$$

$$((a+b)(a+b)(a+b))^{*}(a+b)(a+b)$$

Q: Strings start with a

$$a\cdot(a+b)^{*}$$

aba : $a \cdot \underset{\hookrightarrow b}{(a+b)}\underset{\hookrightarrow a}{(a+b)}$: aba

aaab : $a \cdot \underset{\hookrightarrow a}{(a+b)}\underset{\hookrightarrow a}{(a+b)}\underset{\hookrightarrow b}{(a+b)}$: aaab

Q: Ends with a

$$(a+b)^{*}a$$

Q: Containing a

$$(a+b)^{*}\, a\, (a+b)^{*}$$

Q: Start & End with different symbol $\Sigma=\{a,b\}$

a ———— b or b ———— a

$$a\,(a+b)^{*}\,b \qquad\qquad b\,(a+b)^{*}a$$

$$a\,(a+b)^{*}b \;+\; b(a+b)^{*}a$$

Q: Start and End with same symbol

$$a\,(a+b)^{*}a + b(a+b)^{*}b + a + b$$

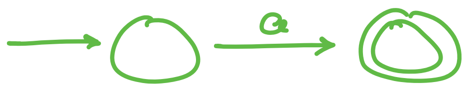
$$L = \{ a, b, aa, aba, aaa \ldots \}$$

CONVERSION of REGULAR EXPRESSION TO FINITE AUTOMATA

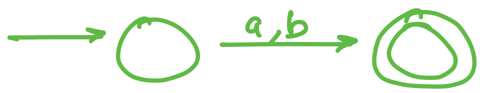
Represent — Acceptor

ϕ Don't accept anything

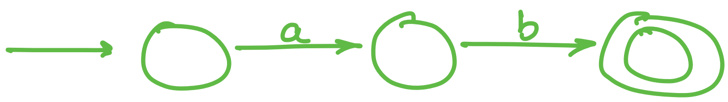
a



$a+b$



$a \cdot b$

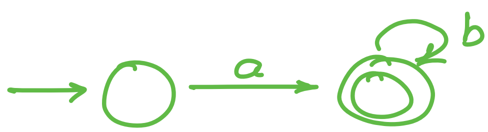


a^*

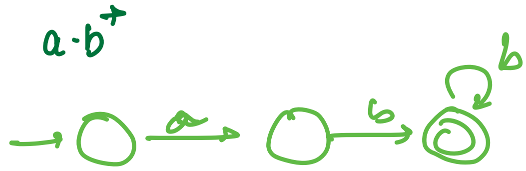
$\hookrightarrow \varepsilon, a, aa, aaa \ldots$

ab^*

$\hookrightarrow a, ab, abb, abbb \ldots\ldots$

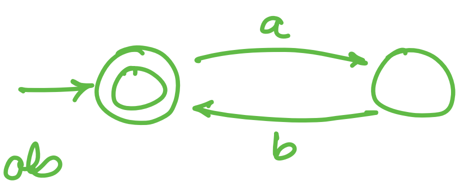


$a \cdot b^+$



$(ab)^*$

$\hookrightarrow \varepsilon, ab, abab, ababab$

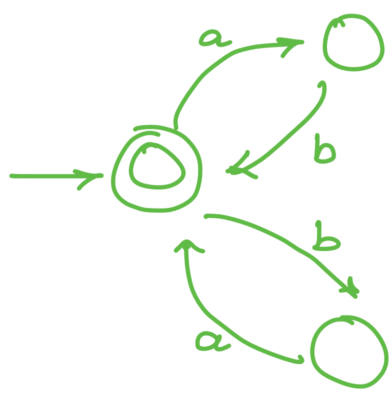


$(ab+ba)^*$

$\hookrightarrow \varepsilon, ab, ba, abba, baab \ldots$

$*: 2$

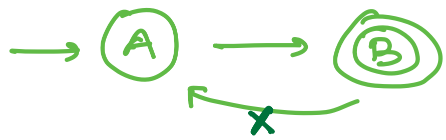
$\underbrace{(ab+ba)}_{ab}\underbrace{(ab+ba)}_{ba} = abba$

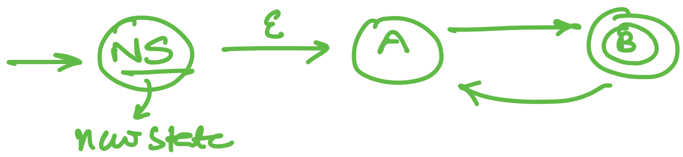


CONVERSION OF FINITE AUTOMATA TO REGULAR EXPRESSION

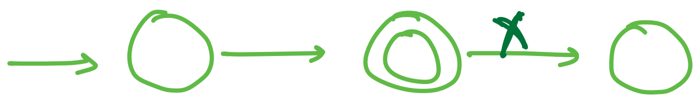
STATE ELIMINATION METHOD

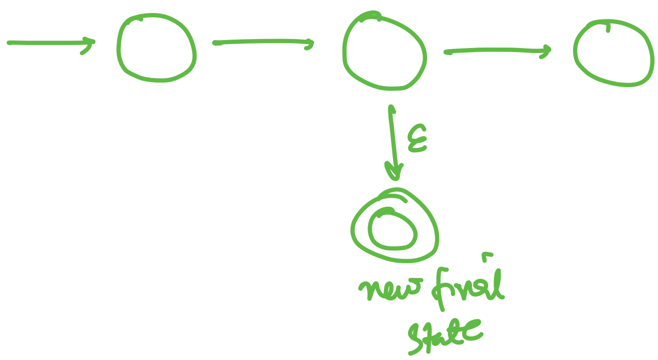
1. Initial State shouldnot have any incoming edge.



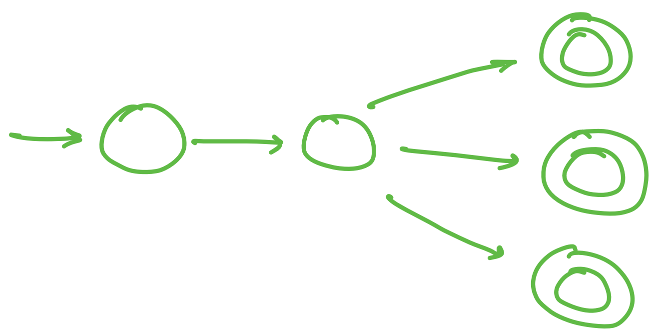


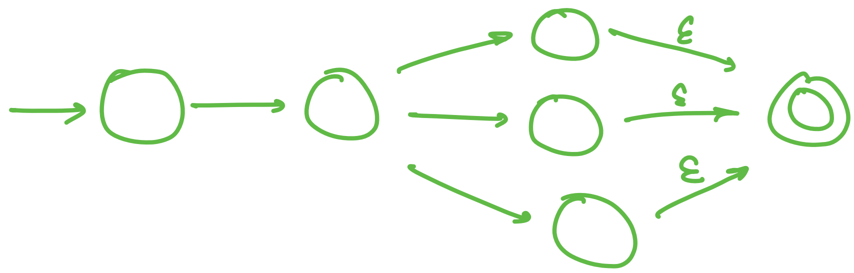
2. Final State shouldnot have any outgoing edge





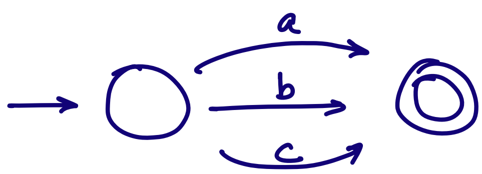
3. One final state

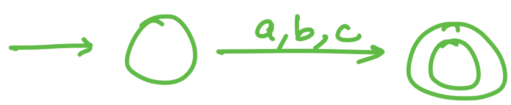




4. Eliminate all states except initial & final state

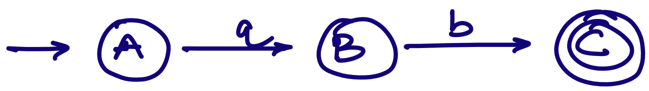
Eg:





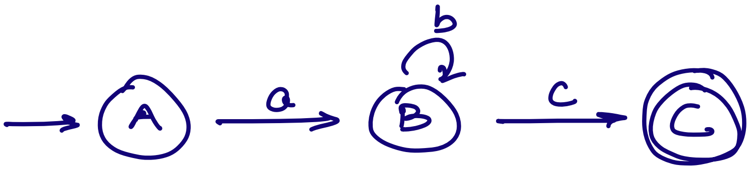
RE: (a+b+c)

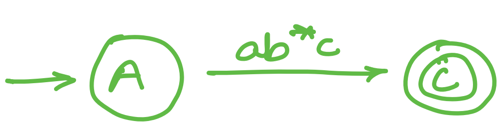
Eg:



RE: a.b

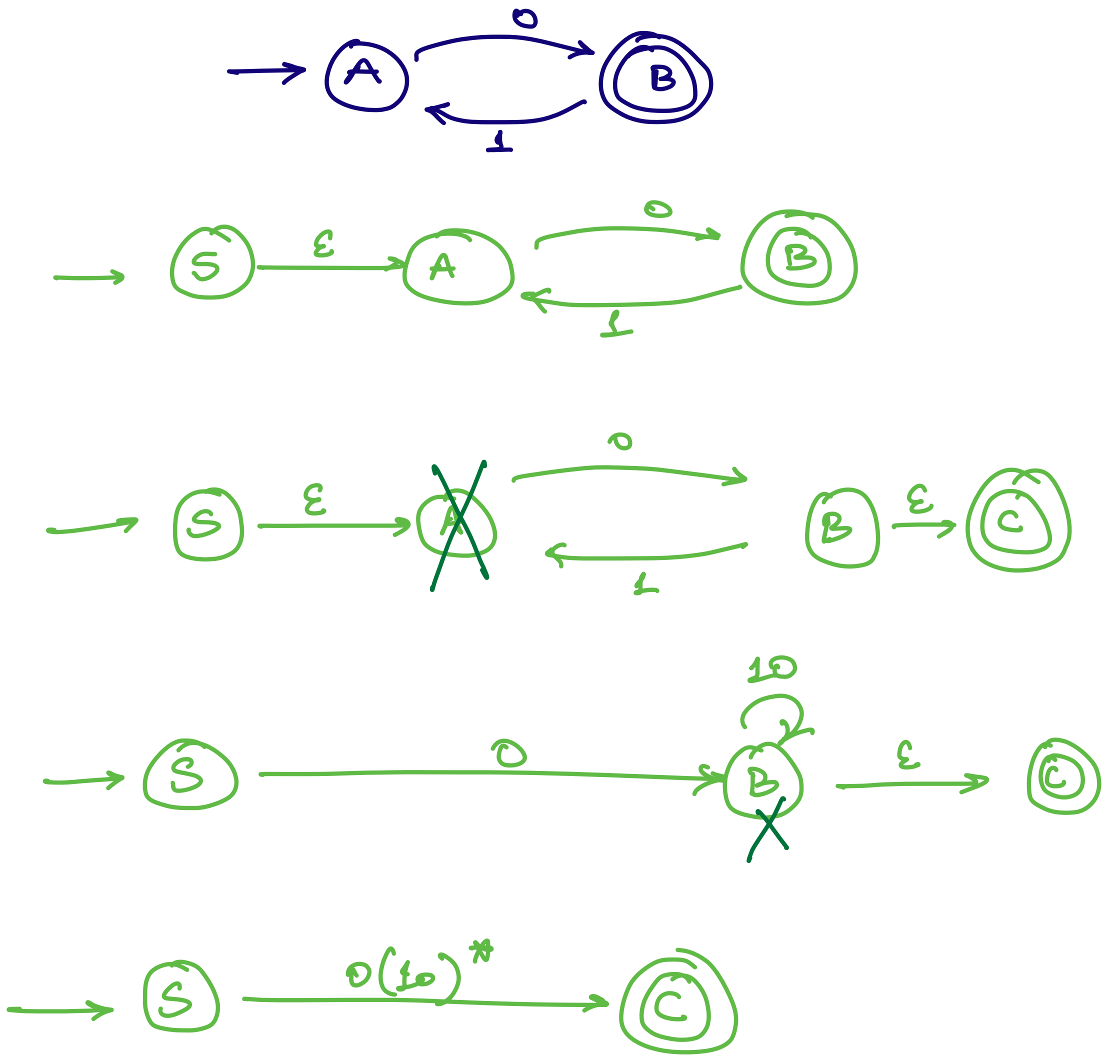
Eg:





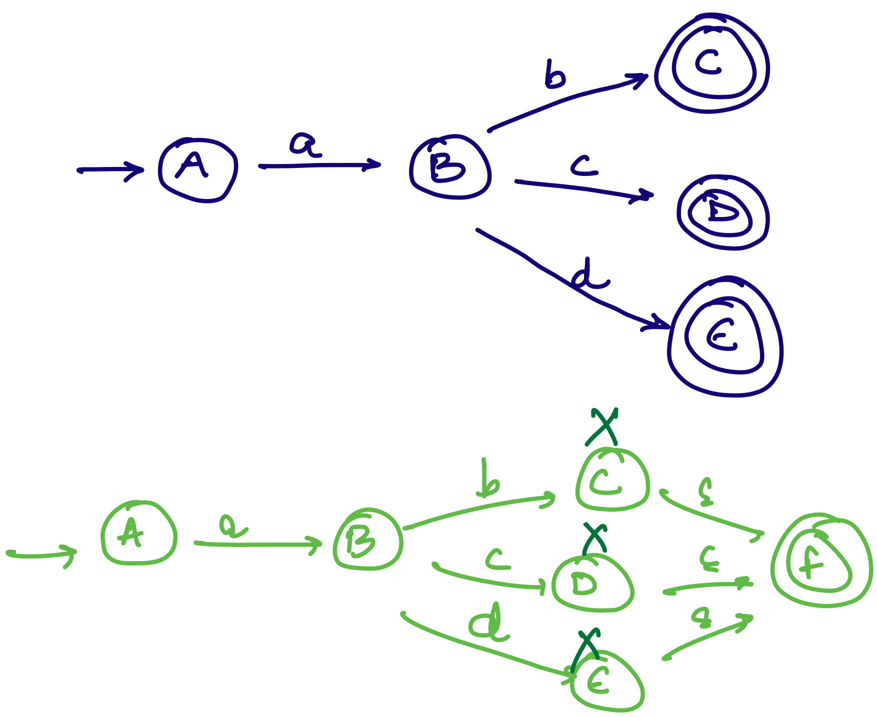
RE: ab*c

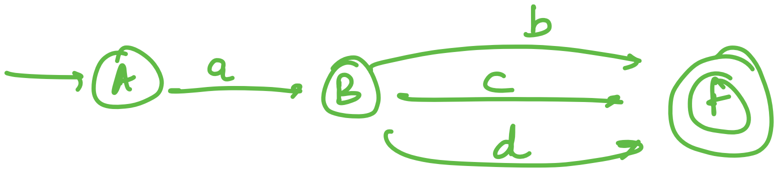
Eg:

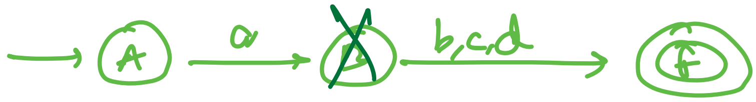


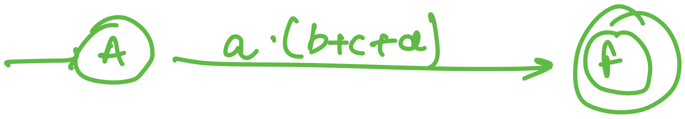
RE: $0(10)^*$

Eg:



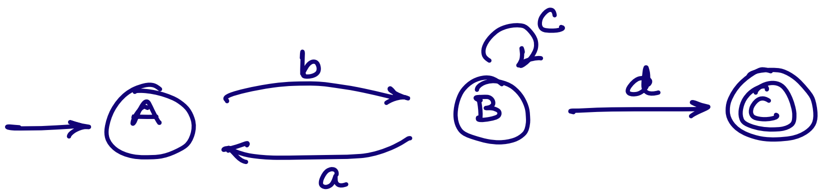


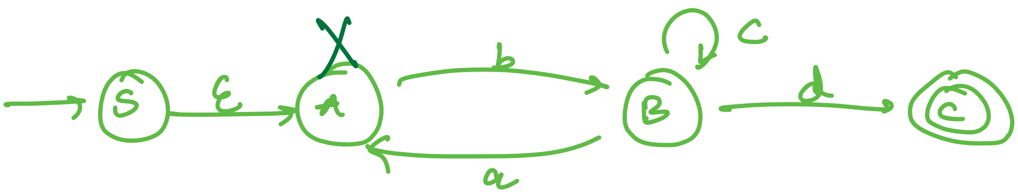


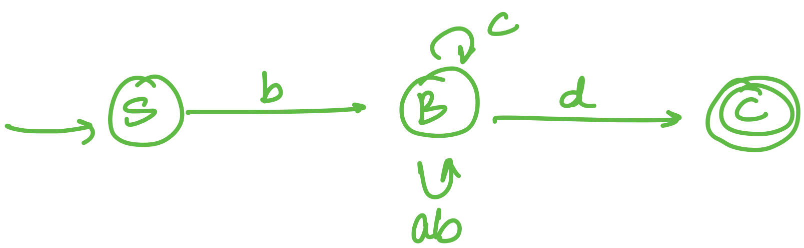


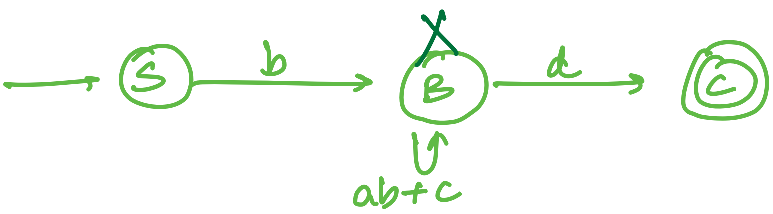
RE: $a\cdot(b+c+d)$

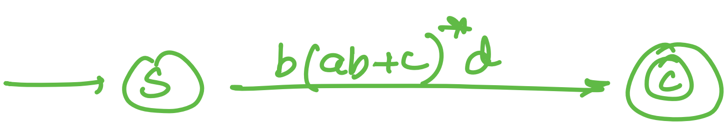
Eg:



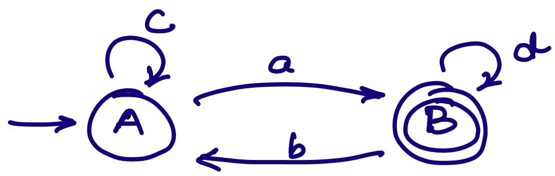


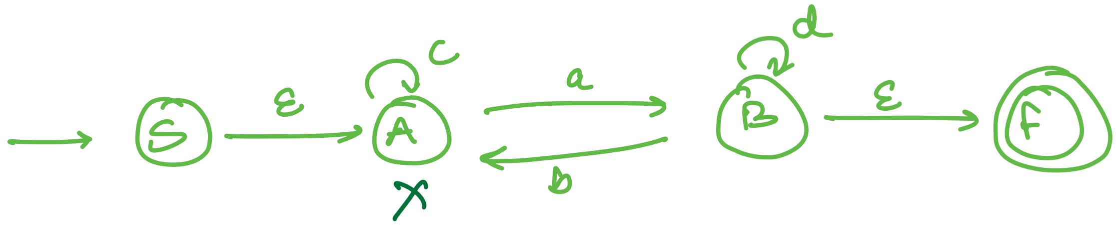


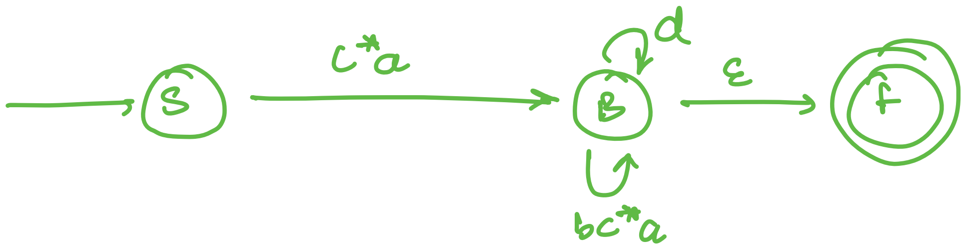


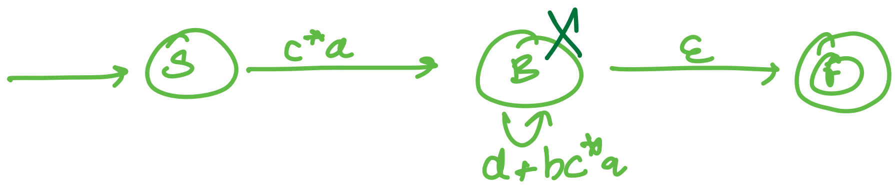


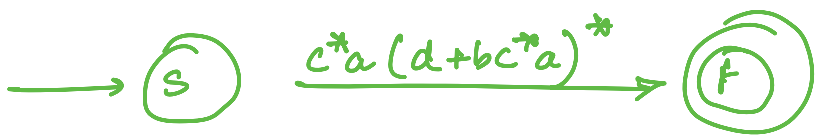
Eg:





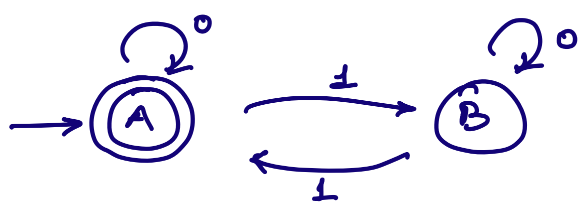




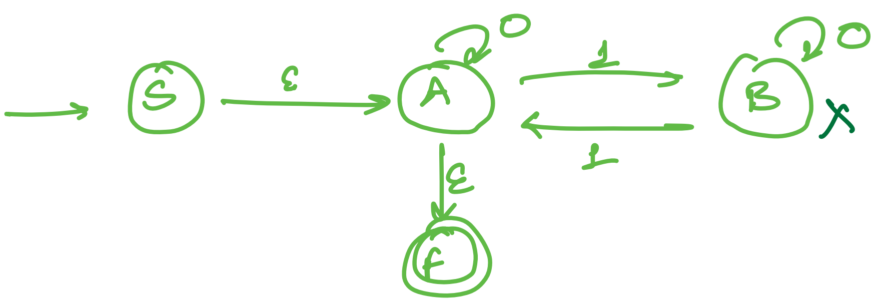


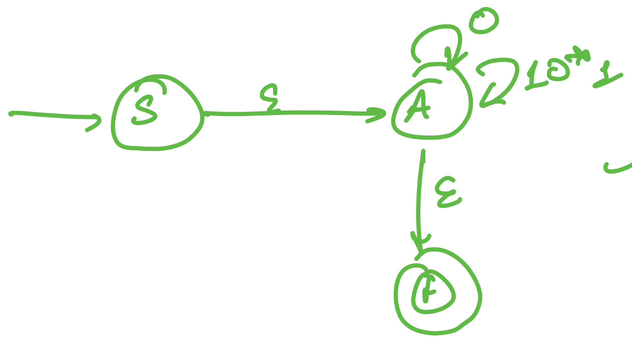
RE: $c^*a(d+bc^*a)^*$

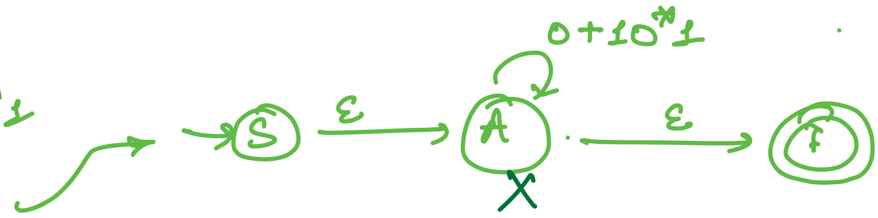
Eg:

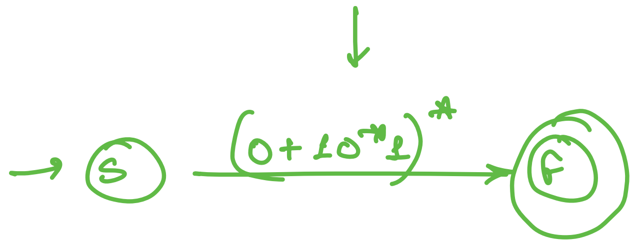


→ even no. of 1's



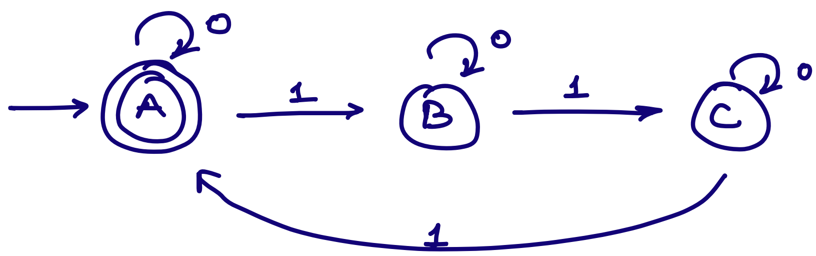






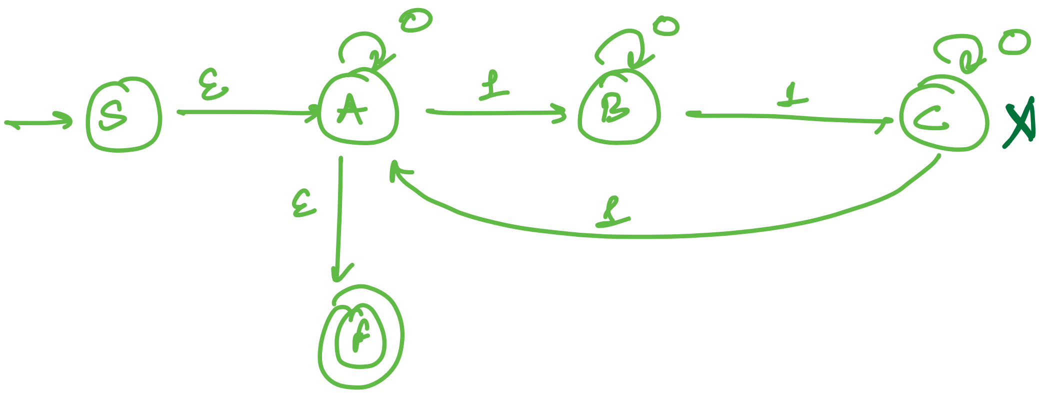
RE: $(0+10^*1)^*$

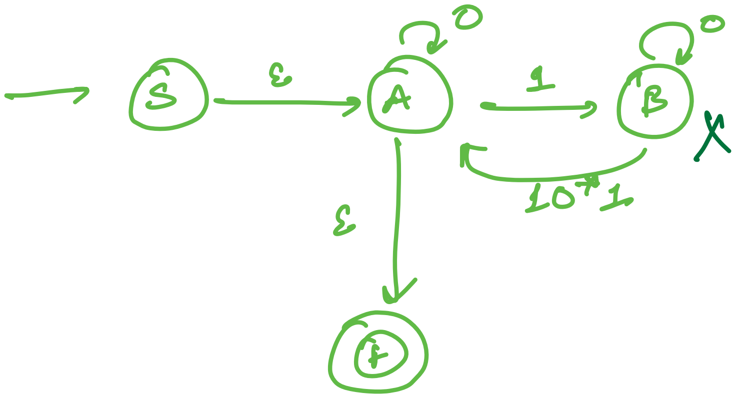
Eg:

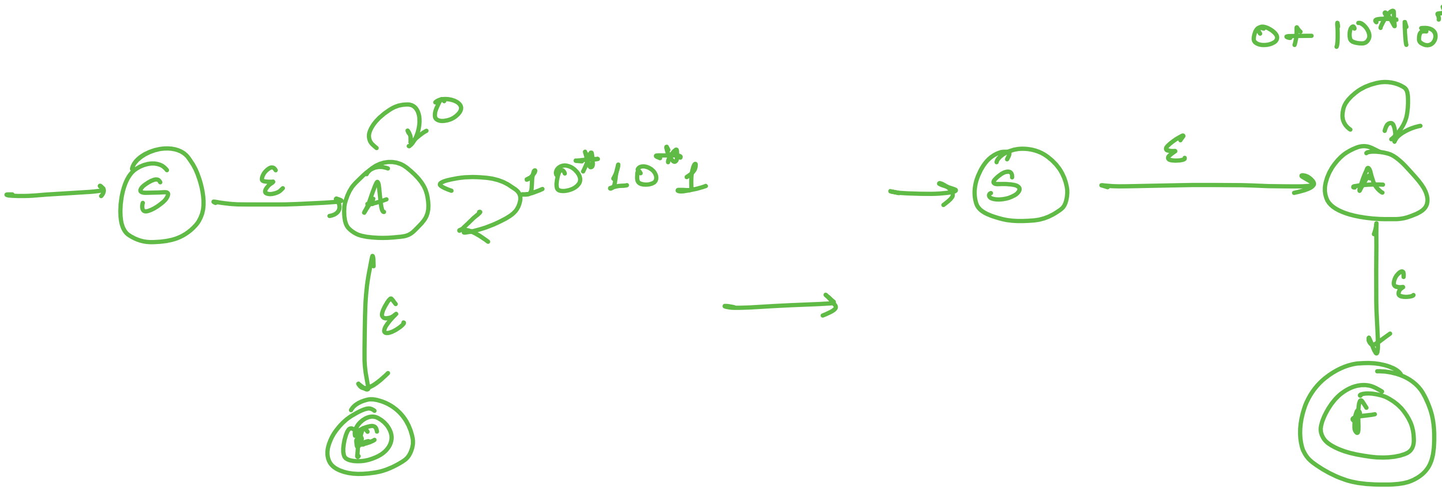


(no of 1's) % 3 == 0

$$\boxed{(0^*10^*10^*1)^* + 0^*}$$







RE: $(0+10^*10^*1)^*$

Kleene's theorem is used to show the equivalence between regular languages, regular expressions, and finite automata. Kleene's theorem states that:

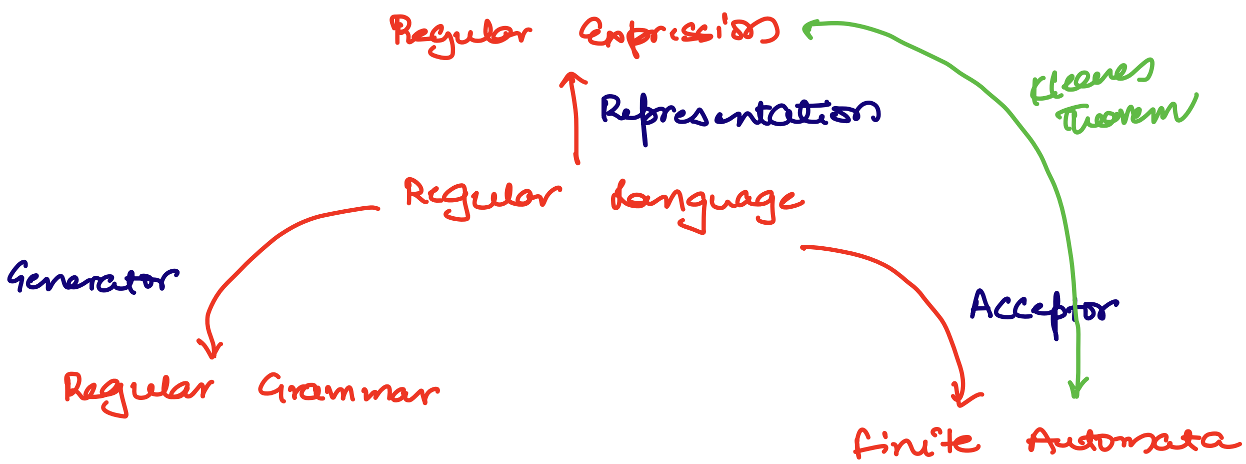
For any regular expression of a language, there exists a finite automaton.

In simple words, a regular expression can be used to represent a finite automaton and vice versa.

$\left.\begin{array}{l} FA \rightarrow RE \\ RE \rightarrow FA \end{array}\right\}$ Equivalent Power

---

Homework: ARDENS THEOREM → find RE of FA.



Testing whether a language is regular or not ?

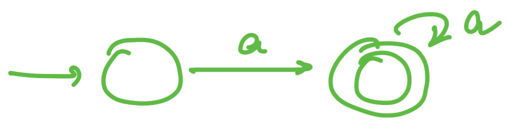
$L \stackrel{?}{=}$ Regular

- language is finite : Regular
- language is infinite : u are able to give a FA or RE then it will be regular language.

eg: $a^n \mid n \geq 1 \longrightarrow a^+$

$L = \{a, aa, aaa, \ldots\}$

FA:



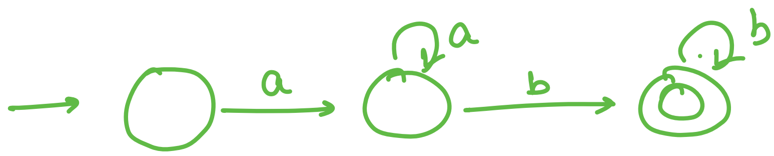
Regular

RE: $a^+ \longrightarrow a \cdot a^*$

eg: $a^n b^m \mid n, m \geq 1$

$L = \{ab, abb, aab, \ldots\}$

FA:



Regular

RE: $a \cdot a^* b \cdot b^*$

eg: $a^n b^n \mid n \leq 10^{10^{10}}$

$n \leq 10^{100}$ → finite

n is bounded, Regular

Eg: $a^n b^n \mid n \geq 1$ ⟶ Not Regular

↳ infinite language

{ab, aabb, $\underbrace{aaa}_{3}\underbrace{bbb}_{3}$, aaaabbbb . . . . }

~~aa^*bb^*~~

↳ abbb, aaab

FA through you can't keep track of count

Eg: $ww^R \mid |w| = 2$ $\Sigma = \{a, b\}$

w	w^R	ww^R
aa	aa	aaaa
ab	ba	abba
ba	ab	baab
bb	bb	bbbb

finite ⟶ Regular

RE: aaaa + abba + baab + bbbb

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

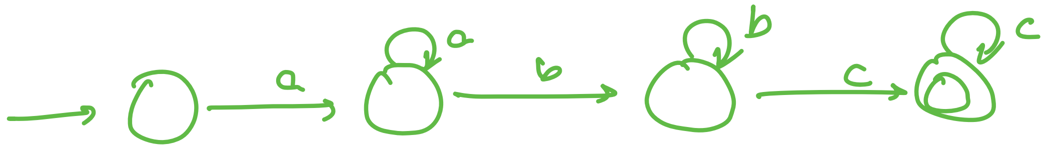
↳ w can be of any length

$\underbrace{abaaa}_{w}$ $\underbrace{aaaba}_{w^R}$

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

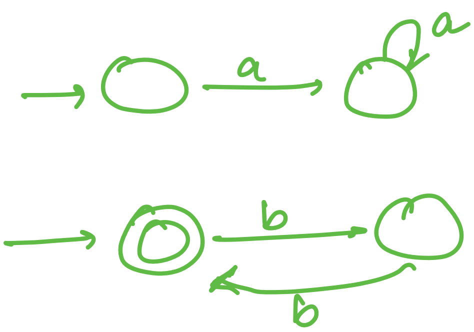
$\underbrace{abaa}_{w}$ $\underbrace{abaa}_{w}$

Eg: $a^n b^m c^k \mid n,m,k \geq 1$ $\longrightarrow$ Regular

FA: 

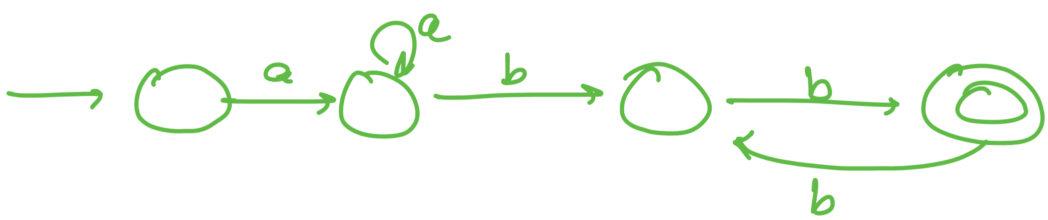
RE: $aa^*bb^*cc^*$

Eg: $a^i b^{2j} \mid i,j \geq 1$ $\longrightarrow$ Regular

FA: 

a^i

b^{2j}



RE: $aa^*(bb)(bb)^*$

Eg: $a^i b^{4j} \mid i,j \geq 1$

RE: $aa^*(bbbb)(bbbb)^*$

Eg: $a^i b^{4j} \mid i,j \geq 0$

RE: $a^*(bbbb)^*$

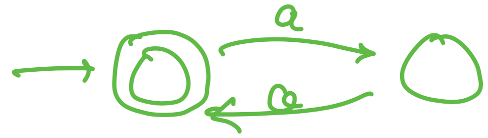
Regular

Eg: $a^n \mid n$ is even $\longrightarrow$ Regular

$L = \{a^0, a^2, a^4, a^6 \ldots\}$

$= \{\epsilon, a^2, a^4, a^6 \ldots\}$ $\longrightarrow$ a^2: create a cycle of 2a's

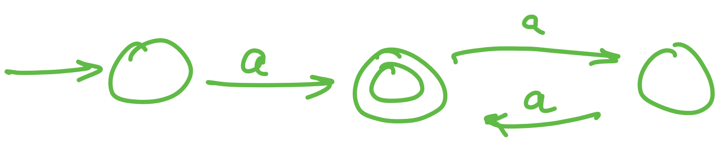
FA:



Eg: $a^n \mid n$ is odd $\longrightarrow$ Regular

$L = \{a^1, a^3, a^5, a^7, a^9 \ldots\}$ cycle of 2a's

FA:



Eg: $a^n \mid n$ is prime $\longrightarrow$ Not Regular

$L = \{a, a^3, a^5, a^7, a^{11}, a^{13}, a^{17} \ldots\}$

no common difference

↓

cannot have a FA