Basics of Theory of computation →

Symbol → Simplest basic non divisible unit of TOC.
a, b, 0, 1 ...

↓

Alphabet → Represented by "Σ", collection of symbols / set of symbols
Σ = {a, b} Σ° → "∈" epsilon, "λ" lambda

↓

String → Collection of alphabets. (Sequence of symbols)
a, ab, aaa, b, bb ...

↓

Language → Collection/set of all string of length (eg → "2")
{aa, ab, ba, bb}

↙ finite ↘ infinite

{ $\Sigma^* \rightarrow (a+b)^*$ ← Klene closure } $\Sigma^\circ = \{\in\}$ empty set
"Infinite language"

{ $\Sigma^+ \rightarrow \Sigma^* - \Sigma^\circ$ ← Positive closure }

Grammer →

Standard way of representing a language.

$G = \{V, T, P, S\}$

- V → Variable
- T → Terminal
- P → Production Rule
- S → Start

- Q : finite set of states.
- Σ : set of input symbols
- q : Initial state
- F : set of final states
- δ : Transition function

FA

- with output
  - → Moore Machine
  - → Mealy Machine
- without output
  - → NFA
  - → DFA
  - → NFA

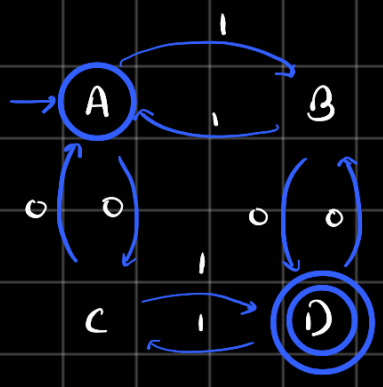
→ (A) initial state

(O) final state

$FA = \{ Q, \Sigma, q_0, F, \$ \}$

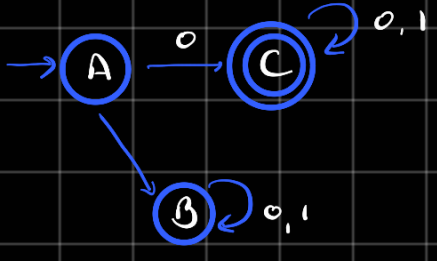
- Q → set of state
- Σ → set of symbol
- q_0 → initial state
- F → Final state
- $\$$ → Transition

DFA → (Deterministic Finite Automata) $Q \times \Sigma \to Q$



	0	1
A	C	B
B	D	A
C	A	D
D	B	C

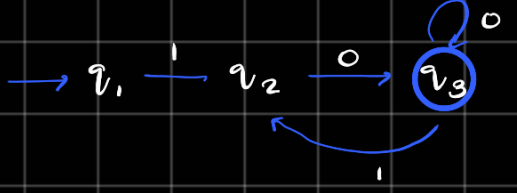
$\Sigma = \{0, 1\}$



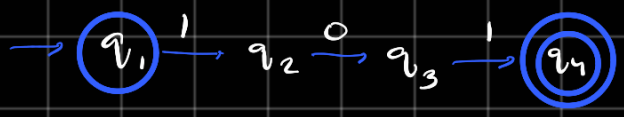
at "1,1" it's at (Trap state)

DFA starts with 1 and ends with 0.

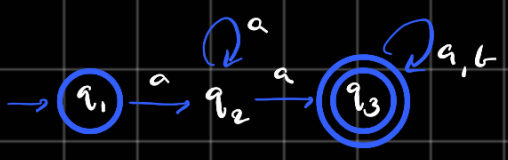
$\Sigma = \{0, 1\}$



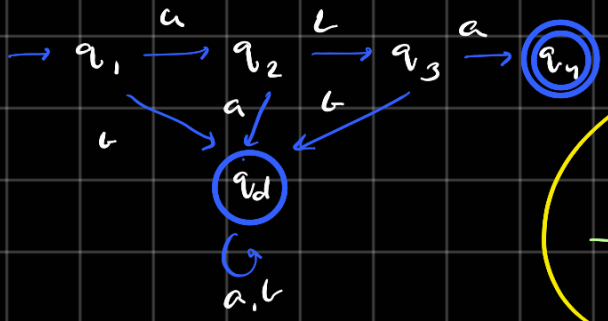
DFD that accepts only string "101", $\Sigma = \{0, 1\}$



DFA that starts with "aa". $\Sigma = \{a, b\}$



→ aba



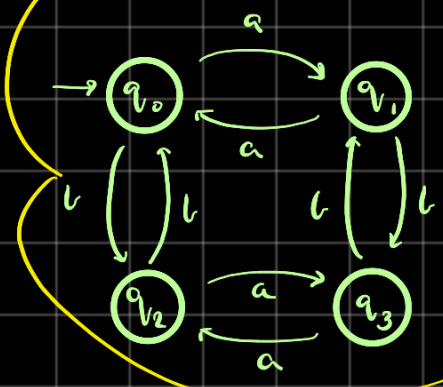
If selected as final state →

q_0 = even a, even b

q_1 = odd a, even b

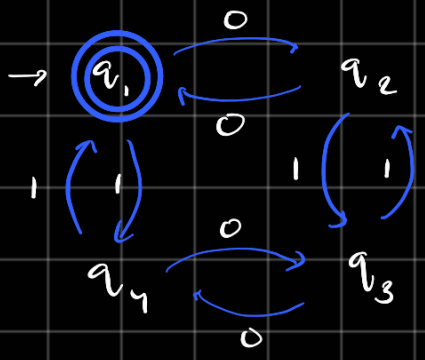
q_2 = even a, odd b

q_3 = odd a, odd b

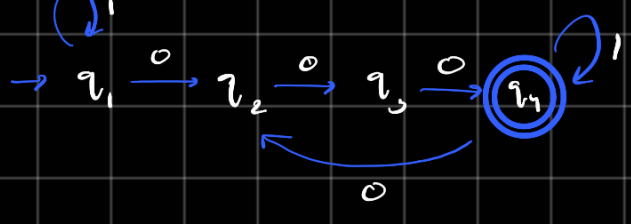


→ even number of 1's and even number of 0's

0011 ✓
1010 ✓
0101 ✓
1100 ✓



DFA that excepts all the strings which have 3 consicutive '0'.
(i.e. 0 will always be in clump of 3.)



Set → collection of distinct objects.

Universal set → U

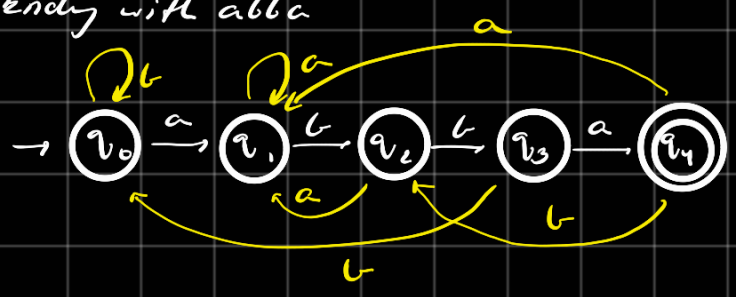
Empty / Null set → { }, ϕ

similarity Difference → {1, 2, 3, 5, 6, 7} A

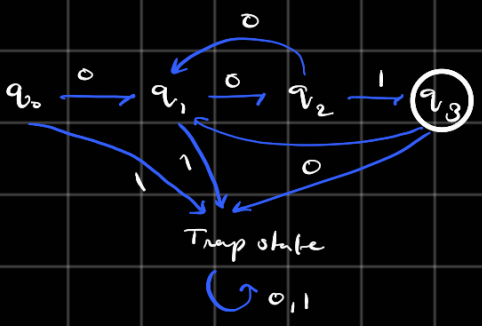
{2, 5, 6, 7, 10, 11} B

$A \oplus B = \{1, 3, 10, 11\}$

→ Ends with "abba"



→ even number of '0' followed by '1'.



001 ✓

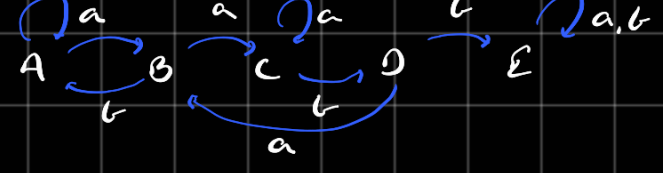
00001 ✓

001001 ✓

0000001 ✓

→ Draw the DFA that does not accepts strings containing aabb in it $\Sigma = \{a, b\}$.

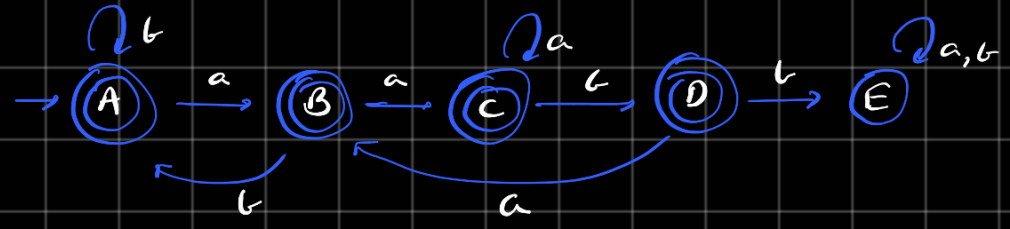
① Create the DFA that accepts abb in it



② Flip the states

final → non final

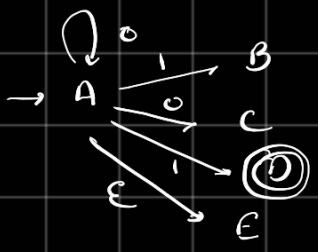
non final → final



NFA → $Q \times (\Sigma \cup \varepsilon) \to 2^Q$ on Q

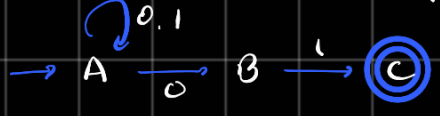
↳ from current state there can be multiple next state.

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



maximum number of state in NFA from one transition can be 2^Q where Q is set of states.

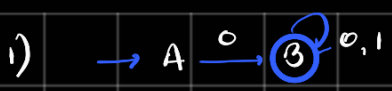
NFA that accepts all strings ending with "01".

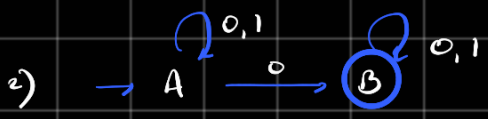


	0	1
A	A,B	A
B	∅	C
C	∅	∅

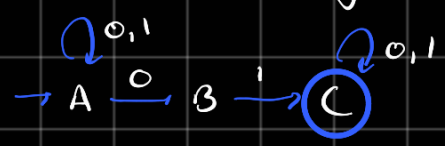
→ NFA having "0111" as substring.

1) Draw the NFA that accepts all strings starting from 0 Σ 0,1.
2) Draw the NFA that accepts all strings containing 0 Σ 0,1
3) Draw the NFA that accepts all strings starting from 10 Σ 0,1
4) Draw the NFA that accepts all strings starting from 01 Σ 0,1

1) 

2) 

Draw NFA containg 01 Σ0,1



	0	1
A	A,B	A
B	∅	C
C	C	C

Conversion from NFA to DFA →

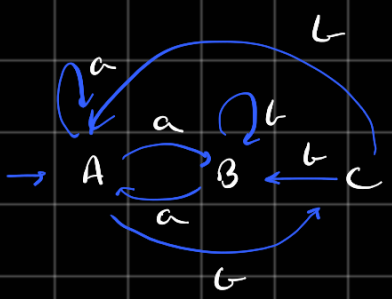
"every DFA → NFA"

but NFA ≠ DFA. But there is equivalent DFA for every NFA.

M = ([A,B,C], (a,b), $, A, {C})

Q Σ Delta Initial Final

$	a	b
A	A,B	C
B	A	B
C	–	A,B



	a	b
A	AB	C
AB	AB	BC
BC	A	AB
C	D	AB
D	D	D

AD → A,B

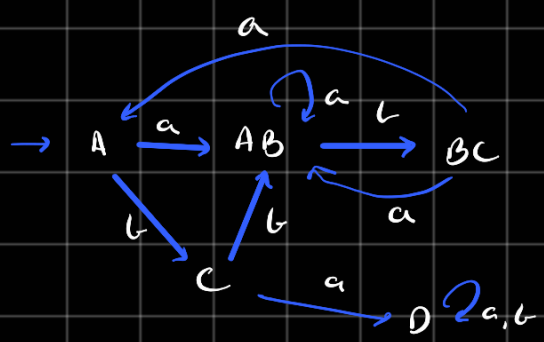
A	A,B	C
B	A	B
	a	b

AB at 'a' ⇒ A, B ∪ A

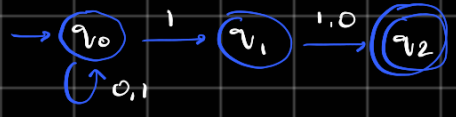
⇒ (A, B)

AB at 'b' ⇒ B ∪ C

⇒ (B, C)



– NFA of all binary strings in which 2nd last bit is 1.

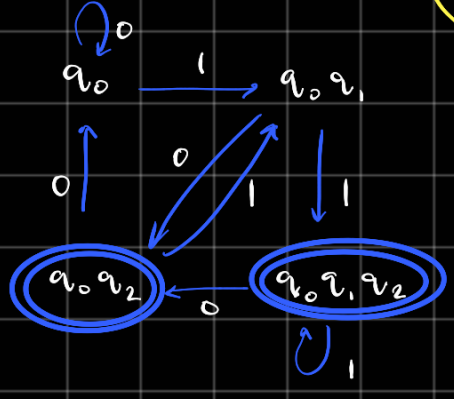


States	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	q_2
q_2	–	–

NFA

To convert it into DFA ⟶

States	0	1
→ q_0	q_0	q_0, q_1
$q_0 q_1$	q_0, q_2	q_0, q_1, q_2
$q_0 q_2$	q_0	q_0, q_1
$q_0 q_1 q_2$	q_0, q_2	q_0, q_1, q_2



Here final states will be all those states that are containing q_2 final state from NFA

NOFA

	0	1
→ q_0	q_1	q_0
q_1	q_0	q_0, q_2
(*) q_2	q_2	q_1

DFA

	0	1
q_0	q_1	q_0
q_1	q_0	q_0, q_2
$q_0 q_2$	q_1, q_2	q_0, q_1
$q_1 q_2$	q_0, q_2	q_0, q_1, q_2
$q_0 q_1$	q_1, q_0	q_0, q_2
$q_0 q_1 q_2$	q_1, q_2, q_0	q_0, q_1, q_2

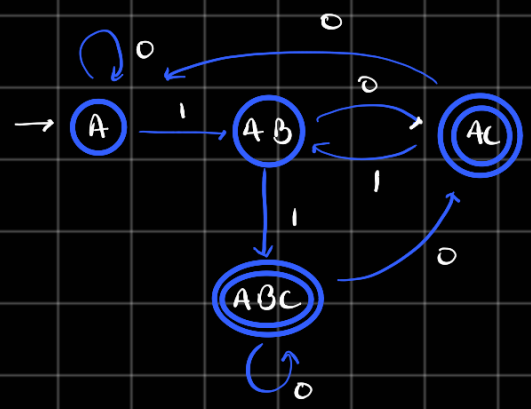
- NFA to DFA

	0	1
A	A	A,B
B	C	C
(*) C	-	-

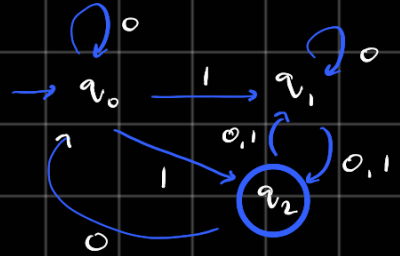
NFA

	0	1
A	A	A,B
AB	AC	ABC
AC	A	AB
ABC	AC	ABC

DFA

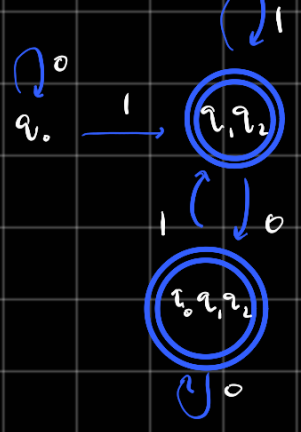


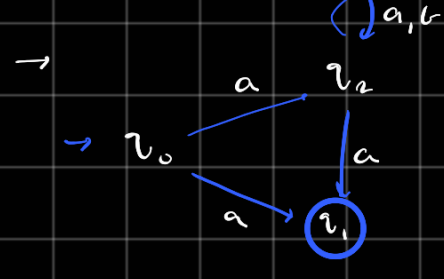
- NFA to DFA



	0	1
q_0	q_0	q_1, q_2
q_1	q_0, q_1, q_2	q_2
(q_2)	q_0, q_1	q_1

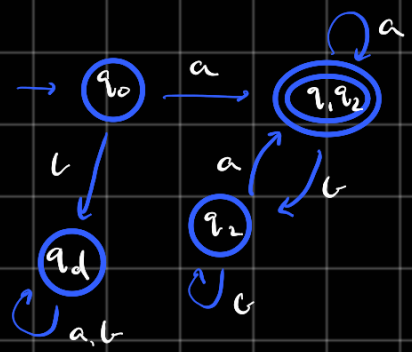
	0	1
q_0	q_0	$q_1 q_2$
$q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$





	a	b
q_0	$q_1 q_2$	—
(q_1)	—	—
q_2	$q_1 q_2$	q_2

	a	b
q_0	$q_1 q_2$	—
$q_1 q_2$	$q_1 q_2$	q_2
q_2	$q_1 q_2$	q_2

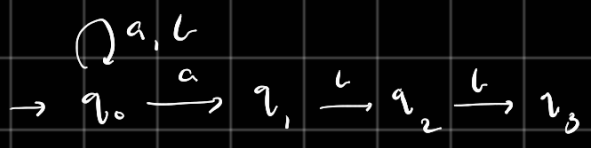


→ "n" number of states in DFA

"m" number of states in NFA

$$1 \leq n \leq 2^m$$

Draw the NFA which accepts all the strings ending 'abb', then draw the equivalent DFA.

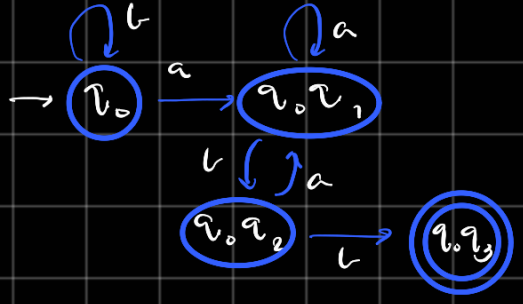


	a	b
q_0	$q_0 q_1$	q_0
q_1	—	q_2
q_2	—	q_3
q_3	—	—

NFA

	a	b
q_0	$q_0 q_1$	q_0
$q_0 q_1$	$q_0 q_1$	$q_0 q_2$
$q_0 q_2$	$q_0 q_1$	$q_0 q_3$
q_3	—	—

DFA



ε-NFA (Epsilon NFA) →

Q	finite set of states
Σ	set of input symbol
q_0	set of initial state
F	set of final state
δ	transition function

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$$

Regular Expression →

Methods to represent a language.

$+ \to$ or $\quad . \to$ and $\quad (a+b) = (a|b)$ (same)

$a+b \to$ either a or b. $\quad a.b \to$ a and b

$R = (11)^* = \{\varepsilon, 11, 1111, \ldots\}$

$R = 1.1^* = \{1\varepsilon, 11, 1.11, 1.111 \ldots\}$

$R = 0+01^* = \{0, 01, 011, 0111 \ldots\}$

$R = 0^*10^*$

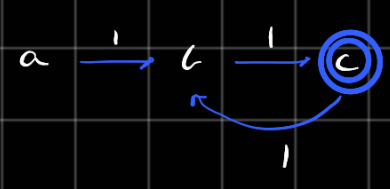
$L = \{\varepsilon 1 \varepsilon, 0.1.0, 0010, \ldots\}$

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

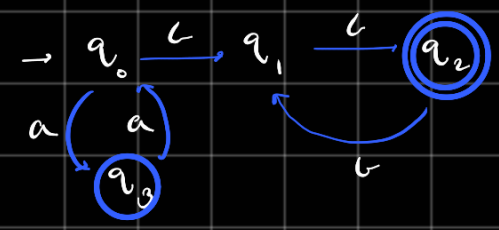
$L = \{\varepsilon abb,$ any combination of $(a+b)$ $abb\}$

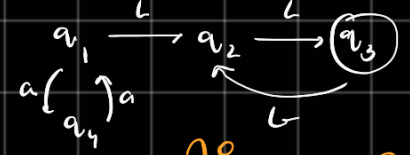
→ Draw the DFA of $R = (11)^*$

→ $R = (11)^+$



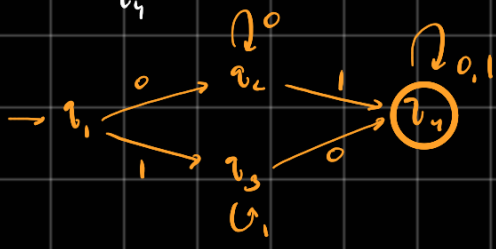
→ $R = (ac)^*(bb)^*bb$





000|1001

1110|1011



- write down regular expression for language L that accepts all strings that have atleast one '0' and one '1'.

$$[(1+0)^*1(1+0)^*0(1+0)^*] + [(1+0)^*0(1+0)^*1(1+0)^*]$$

- write down the regular expression containing a string in which every '0' followed by '11'.

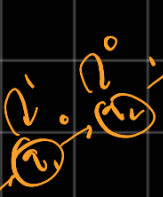
$L = \{1, 11, 1011, \dots\}$

$(1+011)^*$

011, 1, 11, 10111

$\varepsilon, 0, 1, 11$

- write down regular expression that accepts all strings that do not contain the string "01".



1001 ✗ 1110 ✓

101 ✗ 10 ✗

1^*0^*

$q_1 \to q_1 1 + q_3 1 + \varepsilon$

$q_2 \to q_1 0$

$q_3 \to q_2 1$

$q_3 = (q_1 0)1$

$q_1 = q_1 1 + (q_1 0)1 + \varepsilon$

$q_1 = \varepsilon + q_1(1+01)$

$q_1 = \varepsilon.(1+01)^*$

$= (1+01)^*$

$q_1 \to q_1 1 + \epsilon$

$q_2 \to q_1 0 + q_2 0$

$q_3 \to q_2 1 + q_3 0 + q_3 1$

$q_1 \to \epsilon + q_1 1$

$\epsilon.1 =) 1^*$

$q_2 = 1 + q_2 0$

$q_2 = 1^*$

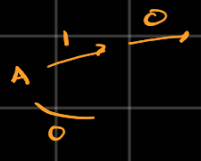
Ardian Theorem

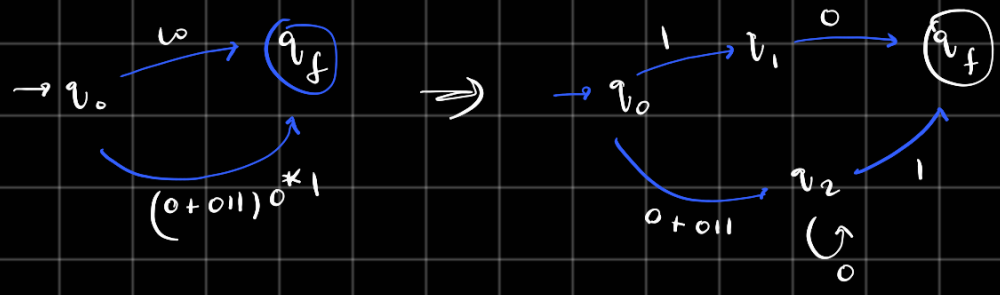
$q_3 = (1^*00^*)1 + q_3(0+1)$

$q_3 = (1^*00^*1)(0+1)^*$

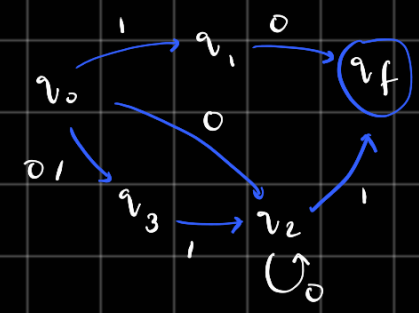
- Draw an NFA for regular expression.

$$\frac{10}{A} + \frac{(0+011)0^*1}{B}$$

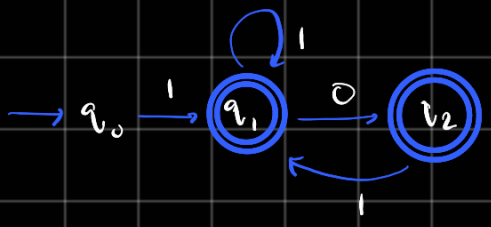




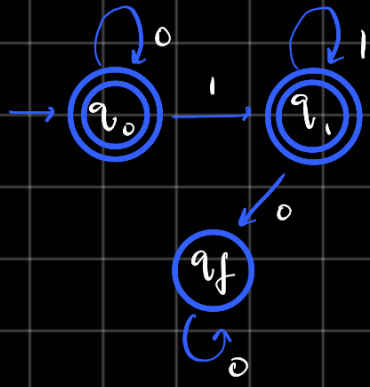
1011

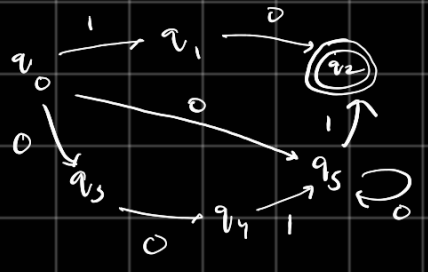


11 110 1101



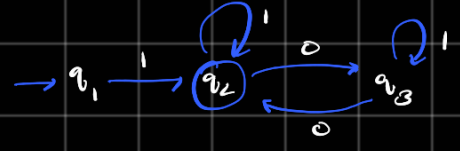
- 0^*1^* 0, 1, 00, 000, 01111,





$\Rightarrow 0^*1 + 10$

- $1(1^*01^*01^*)^*$



- NFA with Λ/ε moves

NFA can make transition without taking any input.

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

$\Rightarrow \varepsilon \subset a \Rightarrow a$

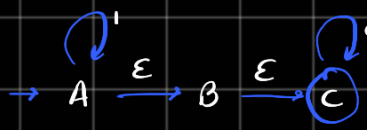
	a
A	D
B	D
C	D

	ε	a	ε
A	A	ϕ	
B	ϕ		
C	D	D	

B	B	ϕ
	C	D D
C	C	D

ε^* → ebsilan closure

ε^* of any state is the set of all states that can be reached from a particular state only by seeing "ε" symbol



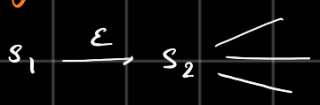
$A = \{A, B, C\}$

$B = \{B, C\}$

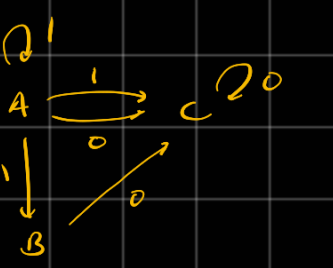
$C = \{C\}$

	ε	0	ε	1	ε
B	B	∅	–	∅	
C	C	C	C	∅	
C	C	C	C	C	∅

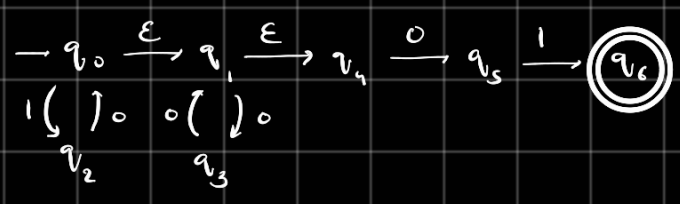
Eleminating ε-Moves →

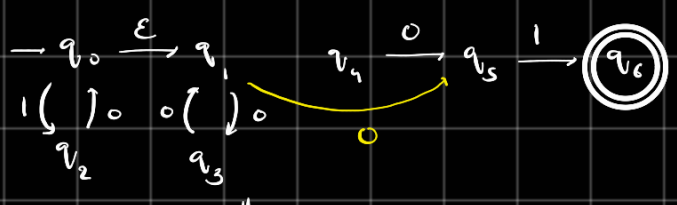


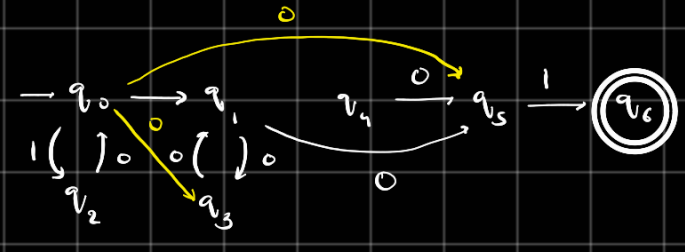
	0	1
A	C	ABC
B	C	–
C	C	

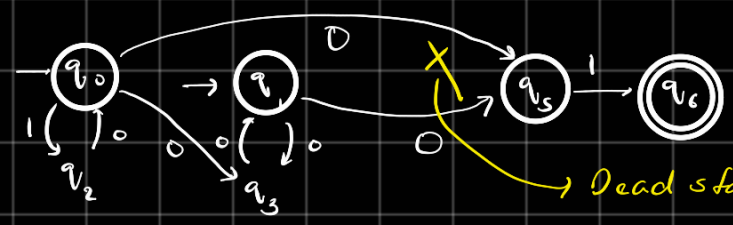


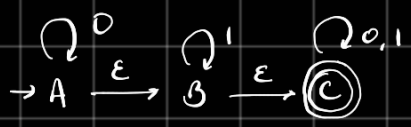
- Find all edges starting from "s_2".
- Duplicate all edges to "s_1" without changing edge lables.
- If "s_1" is initial state, make "s_2" initial.
- If "s_1" is final state, make "s_2" final.



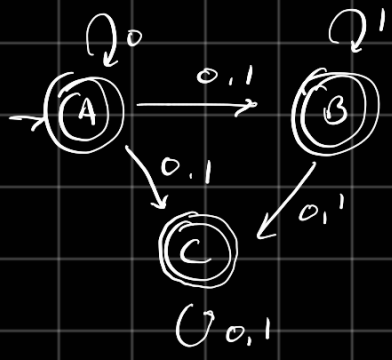








	0	1
A	{A,B,C}	{B,C}
B	C	{B,C}
(C)	C	C



	ε^*	0	ε^*
A	A	A	A B C
B	B	ϕ	–
C	C	C	C

	ε^*	0	ε^*
B	B	ϕ	–
C	C	C	C

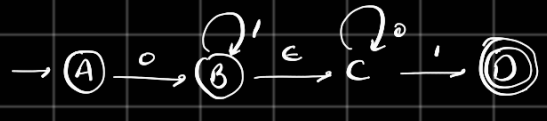
	1	ε^*
A	ϕ	–
B	B	B C
C	C	C

	1	ε^*
B	B	BC
C	C	C

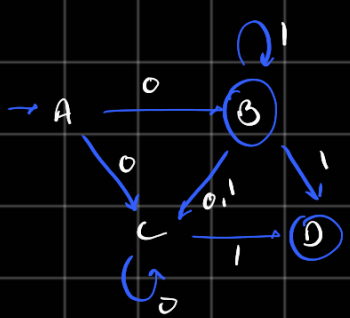
C	ε^*	1	ε^*
	C	C	C

	0	ε^*
C	C	C

– Convert the following ϵ-NFA to its equivalent NFA.



	0	1
A	B,C	–
B	C	B,C,D
C	C	D
(D)	–	–



	ε^*	0	ε^*
A	A	B	B,C
B	B	ϕ	–
	C	C	C
C	C	C	C
D	D	ϕ	–

	ε^*	1	ε^*
A	A	ϕ	–
B	B	B	B,C,D
	C	–	–
C	C	D	D
D	D	ϕ	–

Arden's Theorem →

If P and Q are two regular expression and P does not contains 'ε', then the equation $R = Q + RP$ has the unique solution.

Regular expression

$R = Q + R\underline{P}$
ε

$R = Q + RP$ — (1)

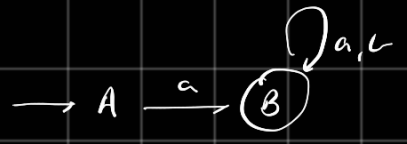
$R = Q + (Q + RP)P$

$= Q + QP + RP^2$

$= Q + QP + (Q + RP)P^2$

$= Q + QP + QP^2 + RP^3$

$\Rightarrow R = QP^*$



↓
Regular exp.

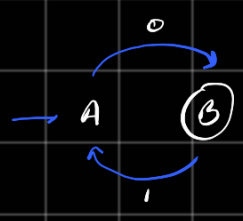
$A = \varepsilon$ — (1)

$B = Aa + Bb + Ba$ — (2)

$B = \varepsilon a + Bb + Ba$

$B = a + B(b + a)$
R Q R P

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



$A \rightarrow B.1 + \varepsilon$ —①

$B \rightarrow A.0$ —②

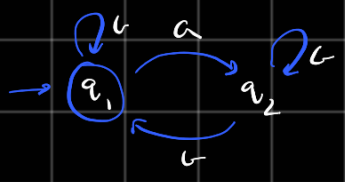
$B \rightarrow (B.1+\varepsilon).0$

$\rightarrow B.1.0 + \varepsilon.0$

$B \rightarrow 0 + B(1.0)$

$R \quad Q \quad R \quad P$

$B = 0.(1.0)^*$



$q_1 \rightarrow q_1 b + q_2 b + \epsilon$

$q_2 \rightarrow q_2 b + q_1 a$

$q_2 \rightarrow (q_1 a) b^*$

$q_1 \rightarrow q_1 b + (q_1 a) b^* + \epsilon$

$\rightarrow q_1 (b + ab^*) + \epsilon$

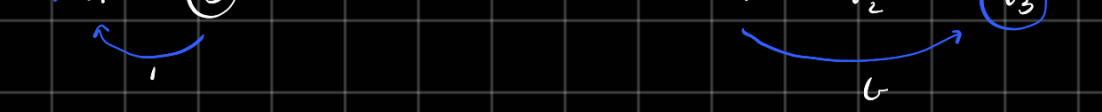
$\rightarrow (b + ab)^*$

sub ② and ③ in ① →

$q_1 = (q_1 a) b + (q_1 b) a + \varepsilon$

$= \varepsilon + q_1 (ab + ba)$

$\Rightarrow \varepsilon.(ab+ba)^* \Rightarrow (ab+ba)^*$



$q_1 \rightarrow \varepsilon$

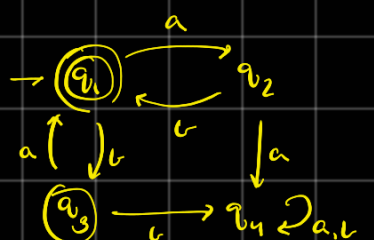
$q_2 \rightarrow q_1 a$

$q_3 \rightarrow q_1 b + q_2 a + q_3 a$

$q_3 \rightarrow \varepsilon.b + \varepsilon.a.a + q_3 a$

$q_3 \rightarrow (b + aa) + q_3 a$

$q_3 = (b+aa)\, a^*$



DFA → Regular expression.

$q_1 \rightarrow \varepsilon + q_2 b + q_3 a$

$q_2 \rightarrow q_1 a$ —②

$q_3 \rightarrow q_1 b$ —③

$q_4 \rightarrow q_2 a + q_3 b + q_4 a + q_4 b$ —④

$R = Q + RP$

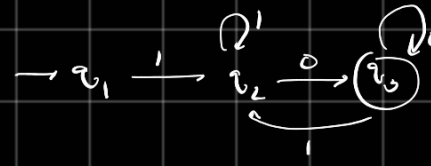
$R = QP^*$ Arden's Theorem

DFA $\Sigma = \{0,1\}$ → 1, 0

1110

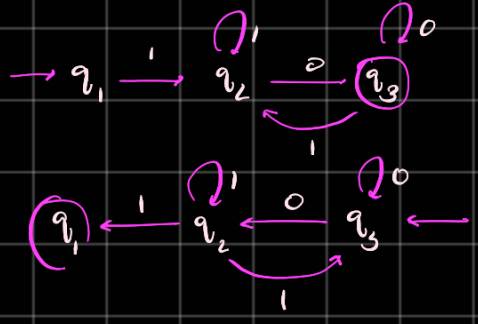
1010

1000



Reversal →

of $L = \{0,1\}$ where start with '1' end with '0'.



① Draw the states as it is.

② Make final initial and initial final.

③ Reverse all edges

④ Loop will remain same

⑤ Remove inappropriate transition state.

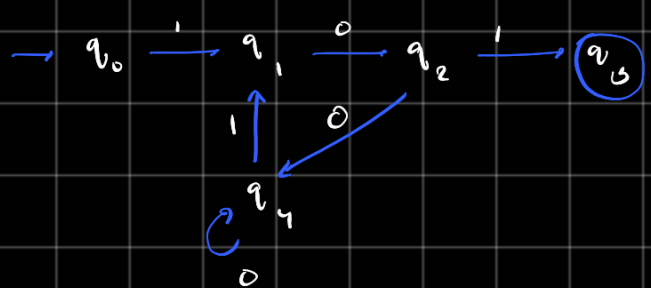
Compliment →

→ Draw a dfa not excepting sub string "101". $\Sigma = \{0,1\}$ Imp (1st MST)

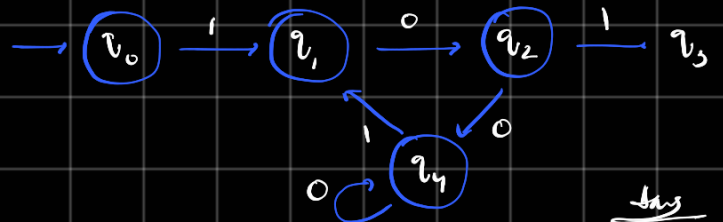
① Dsm DFA accepting '101'

1000101

1001001



② Now make compliment of it { convert final to non final and vice versa }



this

Pumping Lemma →

If L is infinite language then there exists some positive integer 'n' such that any string $w \in L$ has length greater than equal to 'n'.

i.e. $|w| \geq n$ then w can be divided into three parts, $w = xyz$ satisfying following conditions

i) for each $i \geq 0$, $xy^iz \in L$

ii) $|y| > 0$ and

iii) $|xy| \leq n$

- To check language is regular or not.
- It's a negative test.

$w = xyz$

$xy^iz \quad i \geq 0$

$|xy| \leq n$

$y > 0$

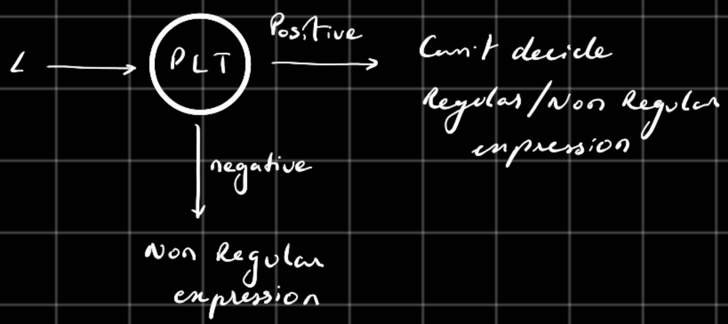
→ $a^n b^{2n}$

$\underline{aa}\ \underline{bb}\,\underline{bb}$
$x \quad y \quad z$

aa bbbb bb $\notin L$ ↳ as when "n = 2"

the language will be formed → aa bbbb

but getting → aa bbbb bb ↳ 2 more 'b'



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

Moore Machine → (FA with output)

$M_o = \{Q, \Sigma, \delta, q_0, \Delta, \lambda\}$

Q → set of states

Σ → set of symbols

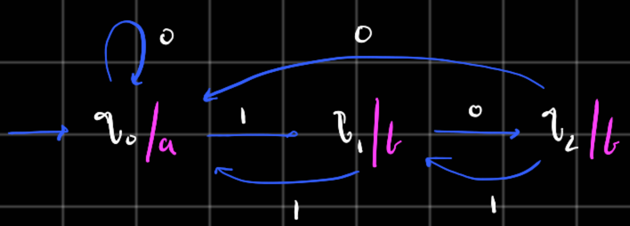
δ → transition function

q_0 → start state

Δ → output symbol

λ → output function

$\lambda : Q \to \Delta$



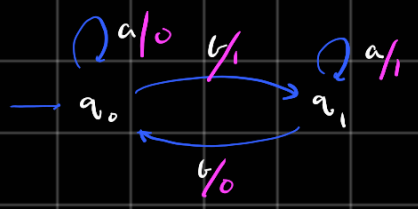
for 'n' inputs there are "n+1" output

Current	Next state		output
	0	1	
q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

⇒ 0 1 1 0 1
a a b a a b

Melay Machine ⟶

$M_e = \{ Q, \Sigma, \Delta, \delta, \lambda, q_0 \}$



Q → set of states
Σ → set of symbols
q_0 → set of initial state
δ → set of transition
Δ → output symbols
λ → output function
$\lambda : Q \times \Sigma \to \Delta$

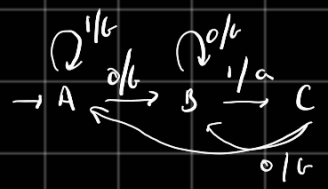
Current state	Next state			
	Input = "a"		Input = "b"	
	state	output	state	output
q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

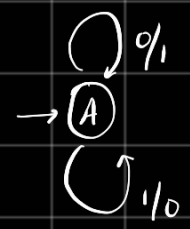
⇒ a b b a
0 1 0 0

8 Construct a melay machine the produces 1's complement of o/p stmg.

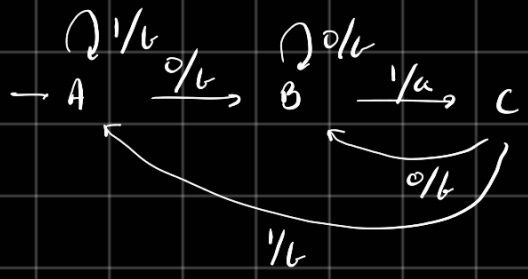
input - 10100

output - 01011





9 Construct a melay machine that produces 'a' when the sequence '01' is incounter.



01 001
a a

8 Design a Melay machine accepting the language consisting of Σ^* where $\Sigma = (a, b)$ and the strg should end with 'aa' or 'bb'

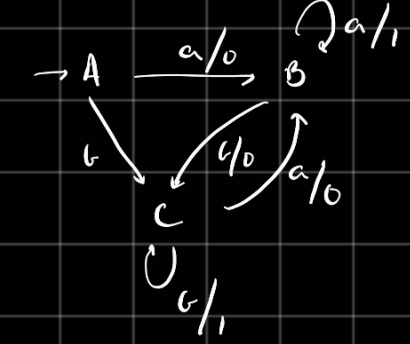
A = 0, 1

so → aa
→ bb } 1

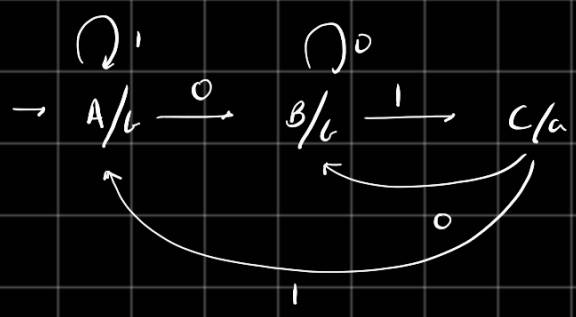
else :- 0

aaa
011
001

baa
abaa
aabbaaa

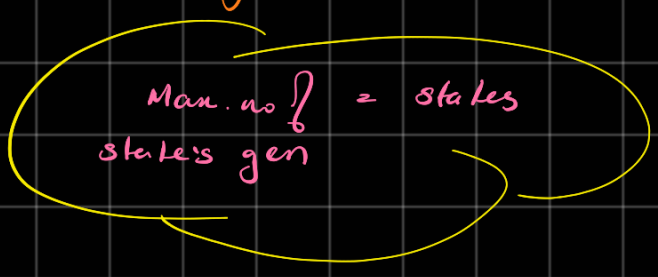


8 Moore machine that prints 'a' when the sequence '01' is encountered.



	0	1	Δ
A	B	A	b
B	B	C	b
C	B	A	a

Moore to Mealy →



number of states = m

number of symbols = n

Q →	a	b	Δ
q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

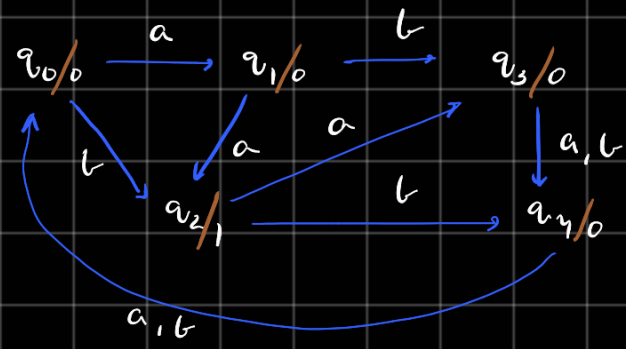
Find the output for

① aabab 001001

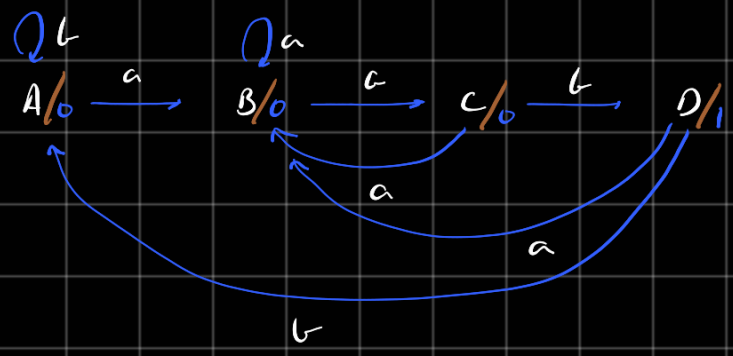
② abbb 00000

③ ababb 000001

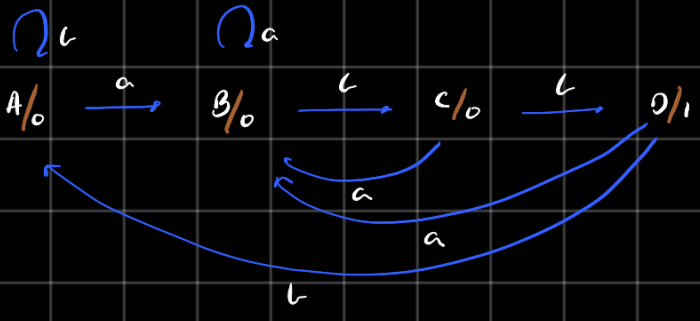
001001



8 Construct a Moore machine that counts the occurrence of 'abb' in the string.



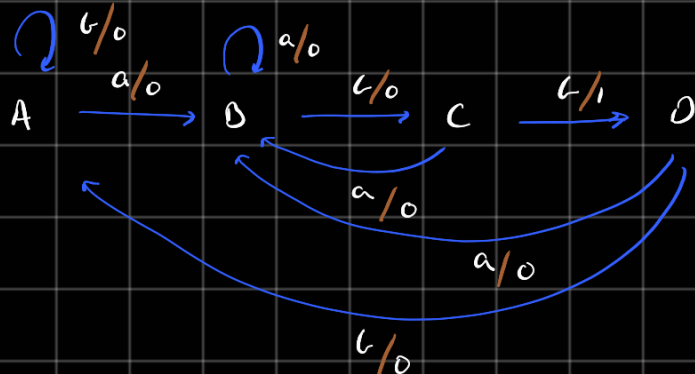
Convert the Moore to Melay method →



	a	b	Δ
A	B	A	0
B	B	C	0
C	B	D	0
D	B	A	1

	a		b	
	T	T̄	T	T̄
A	B	0	A	0
B	B	0	C	0
C	B	0	D	1
D	B	0	A	0

Moore to Melay the number of states remains the same.

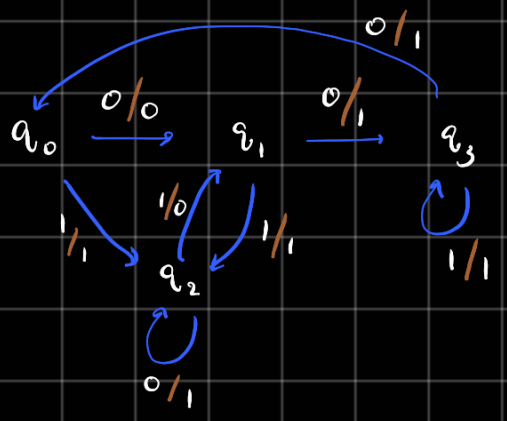


Melay Machine

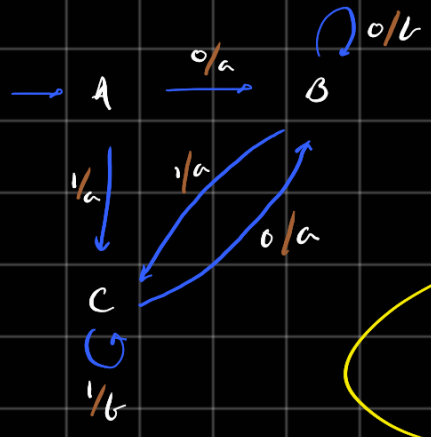
Moorey Transition Table →

	0	1	Δ
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

	0		1	
	T	Σ	T	Σ
q_0	q_1	0	q_2	1
q_1	q_3	1	q_2	1
q_2	q_2	1	q_1	0
q_3	q_0	1	q_3	1



Melay to Moore →

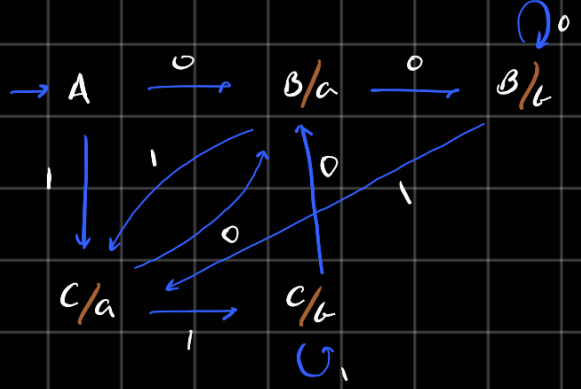


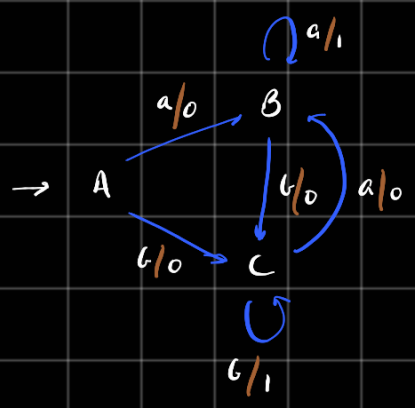
Melay Machine

Max. no of states gen = states × outputs

from Melay to Moore

	0		1	
	T	Σ	T	Δ
A	B	a	C	a
B	B	b	C	a
C	B	a	C	b



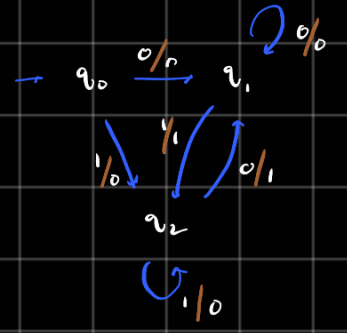


$\Sigma = a, b$

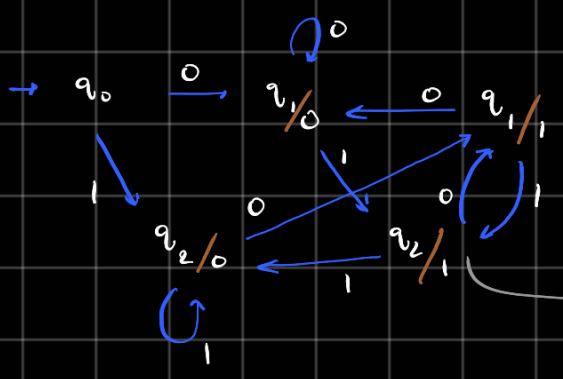
$\Delta = 0, 1$

$Q = A, B, C$

- Example Melay to Moore ⟶



	0		1	
	S	Δ	S	Δ
q_0	q_1	0	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

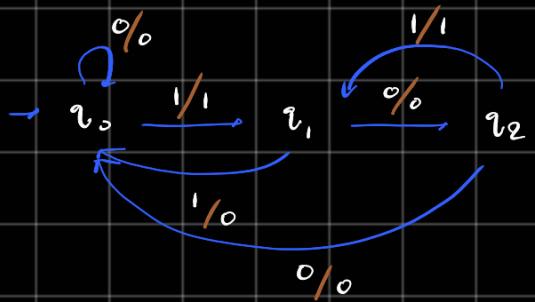


Here similarly ⟶

Here we have created a duplicate state because q_2 have '0' and '1' as output.

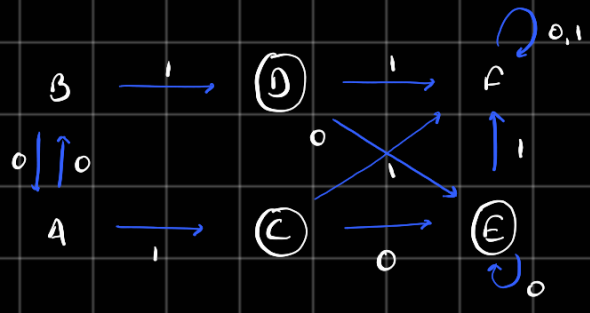
- Example Moore to Melay ⟶

	0	1	Δ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0



Minimization of DFA →

1. Myhill Nerode Theorem →



	A	B	C	D	E	F
A						
B	–					
C	✓	✓				
D	✓	✓	–			
E	✓	✓	–	–		
F	✓	✓	✓	✓	✓	

1. Create Matrix
2. Mark element, different elements. eg :- (Transition, Final) element and not similar eg → (S, S) or (F, F)
3. If unmarked pair (P, Q) such that $[\delta(P,x), \delta(P,x)]$ is marked then mark (P, Q).
4. Combine all unmarked pairs and make single state.

$B,0 \to A$, $A,0 \to B$ } marked $B,1 \to D$, $A,1 \to C$ } marked

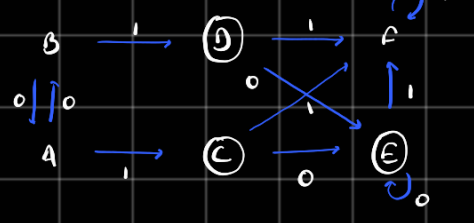
$D,0 \to E$, $C,0 \to E$ } ✗ $C,1 \to F$, $D,1 \to F$ } ✗

$E,0 \to E$, $C,0 \to E$ } ✗ $C,1 \to F$, $E,1 \to F$ } ✗

$E,0 \to E$, $D,0 \to E$ } ✗ $D,1 \to F$, $E,1 \to F$ } ✗

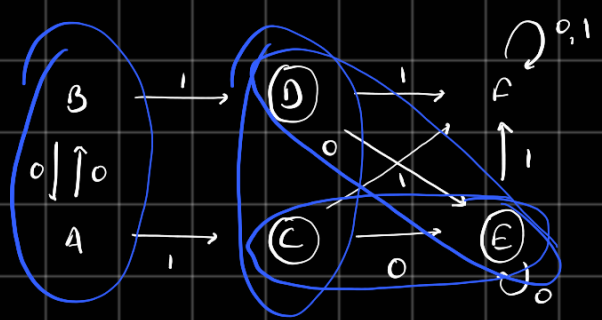
$F,0 \to F$, $A,0 \to B$ } ✗ $A,1 \to C$, $F,1 \to F$ } marked

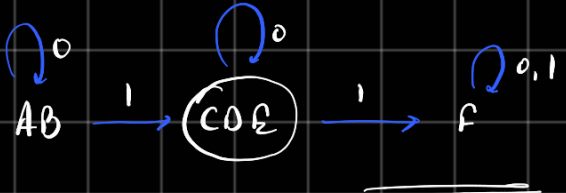
↳ but (F, A) is not so making it.



F, 0 → F
B, 0 → A
} marked

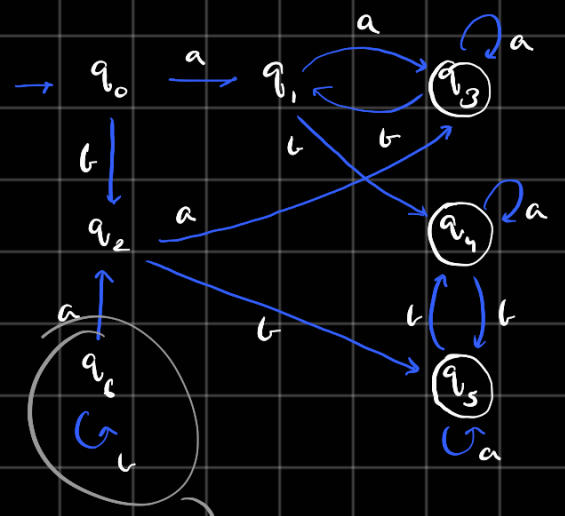
↳ since (F, A) is marked we need to mark (F, B)





---

2. Minimization of DFA / Equivalance of DFA →



	a	b
q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
q_3	q_3	q_4
q_4	q_4	q_5
q_5	q_5	q_4
q_6	q_2	q_6

1) Remove unreachable state

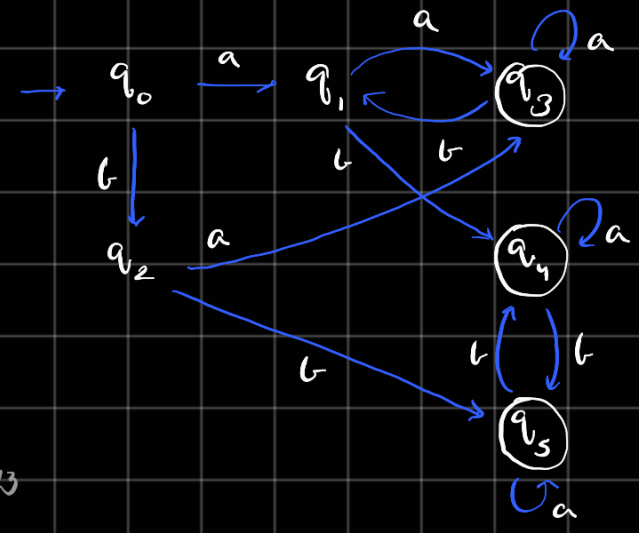
2) Devide into classes

Final Non-final

$\pi_0 = \{q_0, q_1, q_2\}\ \{q_3, q_4, q_5\}$

$\pi_1 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_4, q_5\}$

↳ same for q_2 in q_2, q_3



as q_0, q_1 at 'a', q_1 has out put in different group.

$\pi_1 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_4, q_5\}$

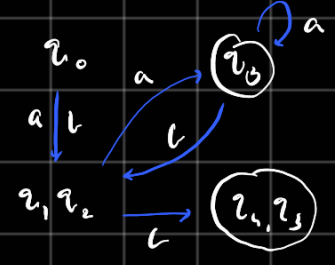
Checking again for new groups →

$\pi_2 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_4, q_5\}$

for example → $q_1, a \to q_3$ $\quad q_1, b \to q_4$

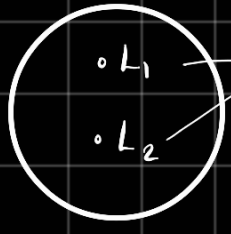
$q_2, a \to q_3$ $\quad q_2, b \to q_5$

going to the same group in each case.



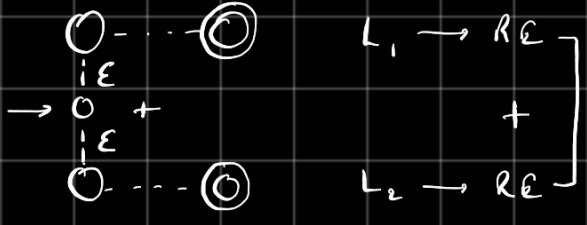
Closure Property of Regular Language →

Regular Language set



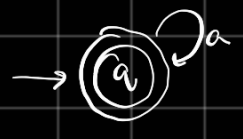
both are part of Regular language

1) Union ($L_1 \cup L_2$)



2) Concatination ($L_1 . L_2$)

3) Closure (*) L^*



$a^* = \{\phi\}, \{a\} \ldots$

4) Compliment $(\bar{L} = \Sigma^* - L)$

5) Intersection $(L_1 \cap L_2 = \overline{\bar{L_1} \cup \bar{L_2}})$

6) Difference $(L_1 - L_2 = L_1 \cap \bar{L_2})$

7) Reversal $(L)^R$

$ab \rightarrow ba$

8) Homomorphism (substitution function)

9) Reverse Homomorphism

10) Quotent operation

11) DNQT

12) Substitution

13) Infinite Union *

Homomorphism ⟶

$h(L) : \{h(\omega) \mid \omega \in L\}$

where $h : \Sigma$ to Γ^* so called Homomorphism

(Kow star)

$\Sigma = \{0,1\}$ $\Gamma = \{a,b\}$

$h(0) = aa$ $h(1) = bb$

so, if $L = \{00, 101\}$

Then find $h(L) \rightarrow \{aaaa, bbaabb\}$

$(0+1)^*1$

$(aa+bb)^*bb$ } ← Regular expression of homomorphism

Inverse Homomorphism ⟶ $h^{-1}(L)$

Let $h(0) = a$, $h(1) = b$, $h(2) = ab$

$\Sigma = \{0,1,2\}$, $\Gamma = \{a,b\}$

let $L = \{abab\}$

$h^{-1}(L) = \{0101, 201, 012, 22\}$

$h(h^{-1}(L)) = \{abab, abab, abab, abab\}$

→ $h(0) = aa$, $h(1) = bb$

$\Sigma = \{0,1\}$, $\Gamma = \{a,b\}$

$L = \{aa, aabb, baab, ababba\}$

$h^{-1}(L) = \{0, 01, \times, \times\}$

$h(h^{-1}(L)) = \{aa, aabb\}$

sub set of the language (L)

Quotent operation ⟶

- Left Quotent (cut Prefix)
- Right Quotent (cut Suffix)

$\frac{L_1}{L_2} = \{x \mid xy \in L_1, \text{ for some } y \in L_1\}$ Right quotient

$\frac{L_1}{L_2} = \{y \mid xy \in L_2, \text{ for some } x \in L_2\}$ Left quotient

$L_1 = \{10, 100, 1010, 101110\}$

$L_2 = \{10\}$

Right → $\frac{L_1}{L_2}$ $\frac{10}{10}, \frac{100}{10}, \frac{1010}{10}, \frac{101110}{10}$

$\Rightarrow \{\varepsilon, \varepsilon, 10, 1011\}$

Left → $\frac{L_1}{L_2}$ $\frac{10}{10}, \frac{100}{10}, \frac{1010}{10}, \frac{101110}{10}$

$\Rightarrow \{\varepsilon, 0, 10, 1110\}$

→ $L_1 = a^*b$

$L_2 = ab$

$L_1 = \{b, ab, aab, aaab, \ldots\}$

$L_2 = \{ab\}$

Left ⇒ $\frac{L_1}{L_2} = \frac{b}{ab}, \frac{ab}{ab}, \frac{aab}{ab}, \ldots$

$\Rightarrow \{\varepsilon, \varepsilon, \varepsilon, \ldots\}$

Right ⇒ $\frac{L_1}{L_2} = \frac{b}{ab}, \frac{ab}{ab}, \frac{aab}{ab}, \frac{aaab}{ab}, \ldots$

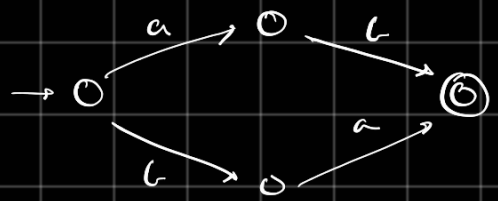
$\Rightarrow \{\varepsilon, \varepsilon, a, aa, aaa, \ldots\}$

Init operation (Initial / Prefix) →

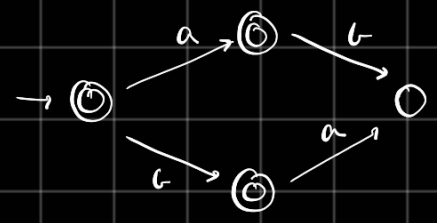
{set of all Prefix of $\omega \in L$}

$L = \{ab, ba\}$

prefix(ab) = ε, a, ab
prefix(ba) = ε, b, ba
→ union → Init {ε, a, ab, ba, b}



$L = \{ab, ba\}$



Init = {ε, a, ab, b, ba}

∴ Regular language is closure under INIT

Infinite Union →

$L_1 = \{ab\}$

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

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

⋮

$L_n = \{a^nb^n\}$

Regular expression are not closed under Infinite Union

$L_n = \{a^nb^n\}$, $n \geq 1$

↑

It is not a regular language

Content free grammer →

Is a formal grammer which is used to generate all the possible patterns of strings in a given formal language

$G = (V, T, P, S)$ Set of recursive rules used to generate patterns of strings.

V → Variables (in capital letter)

T → Terminal (in small letter)

P → Production

S → start

→ a^*

$S \rightarrow aS \mid \varepsilon$

→ $(0+1)^*$

(Regular expression to content free grammer)

Content free language →

Content free language is generated by content free grammer

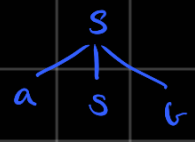
- set of CFL is identical to set of language accepted by pushdown automata. and set of regular language is a subset of CNL.

CFL → CFG

1) $a^n b^n$, $n \geq 0$

$S \rightarrow aSb$

$S \rightarrow \varepsilon$



2) $a^n b^{n+2}$

$S \rightarrow aSb \mid bb$

3) $a^m b^n$, $m > n$, $n \geq 0$

$S \rightarrow A.B$

$A \rightarrow aA/a$

$B \rightarrow bB/\varepsilon$

$a^n b^n$

aaaabb

$S \rightarrow aSb \mid bb$

$S \rightarrow A.B$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

$a^{2n} b^n \quad n=0$

aab

aaaabb

$S \rightarrow A.B$

$A \rightarrow aA$

4) $a^{2n} b^n$, $n \geq 0$

aaaabb
aaab

$S \rightarrow aaSb / \varepsilon$

5) $\{ w \mid n_a(w) = n_b(w) \}$

$S \rightarrow aSb / bSa / \varepsilon$

→ Not getting abba

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

Derivation ⟶ (Drivation Tree)

Deriving the string by following the grammer.

1) Left Most Derivation
2) Right Most Derivation

$S \rightarrow AB$

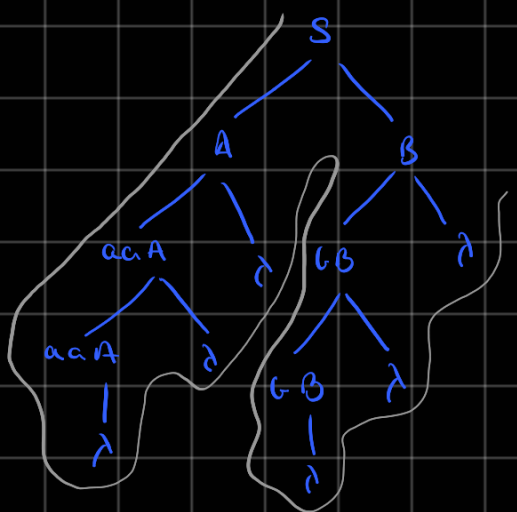
$A \rightarrow aaA / \lambda$

$B \rightarrow bB / \lambda$

check weather $\omega = aaaabb \in L(G)$

A A
aa aa

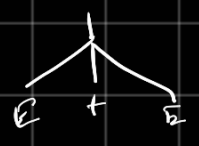
LMD ⟶



$aa\,aa\,bb \in L(G)$

$E \rightarrow E + id$

$E \rightarrow E + E$



$a^{2n} b^n$ LMD

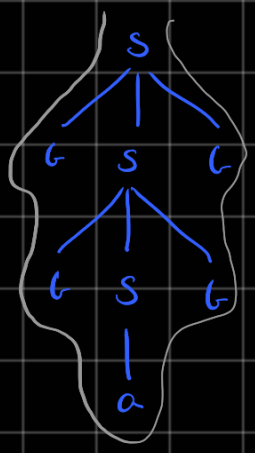
$A \rightarrow aaA / \epsilon$

$B \rightarrow bB / \epsilon$

- Draw the derivation tree

$S \rightarrow bSb / a / b$

$bbabb \in L$



bbabb ∈ L

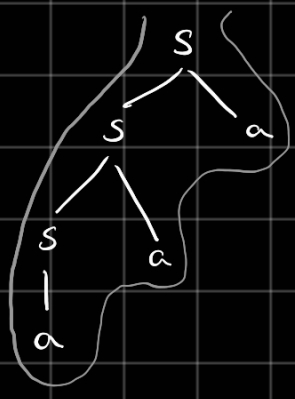
Ambiguous →

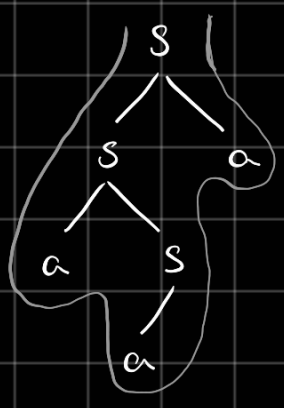
There exists more than one derivation for any word that belongs to Grammer.

$S \to Sa \mid aS \mid a$

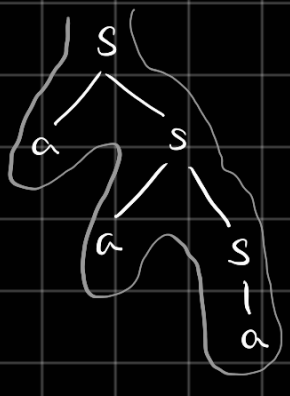
$w = aaa \in L$

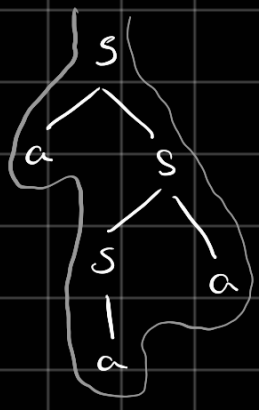
LMD →





RMD →





Unambiguous →

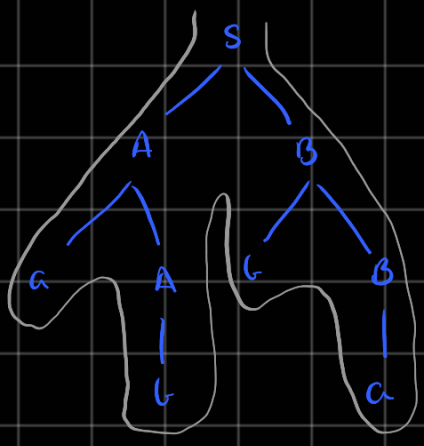
Their exists exactly one derivation for any word that belongs to Grammer

$S \to AB$

$A \to aA \mid b$

$B \to bB \mid a$

$w : abba$



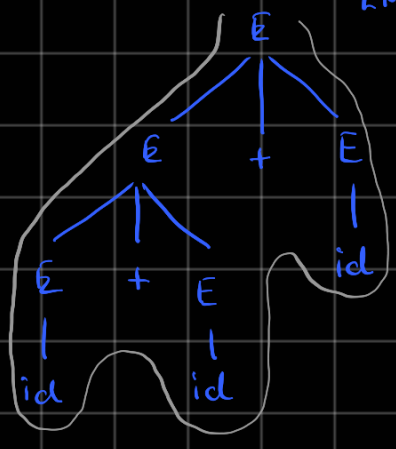
→ abba

unambiguous →

$E \rightarrow E + E \mid id$

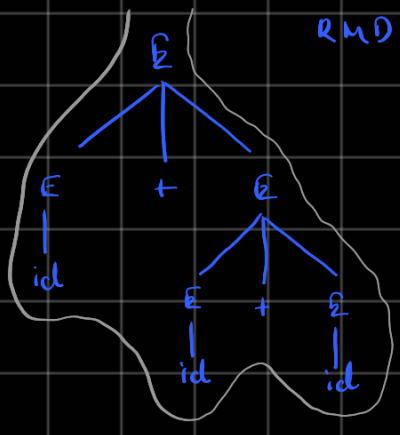
w : id + id + id

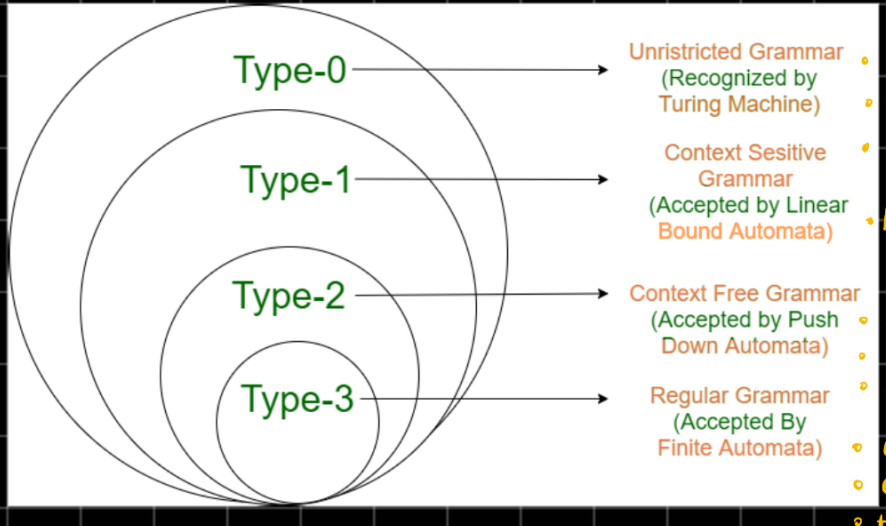
LMD



Ambiguous →

RMD





- Software testing
- Digital computer
- Machine learning
- Parser tree
- Stack application
- Arithmetic operation
- Syntax Analysis
- Media and Money
- Compiler designing
- Text editor

Regular Grammer →

A grammer can be described using 4 tuples $G = (V, T, P, S)$

V → Variable

T → Tuple

S → Start

P → Production

Regular Grammer

Right Linear Grammer

$A \to xB$

$A \to x$

where $A, B \in V$, $x \in T$

eg $\to ab S \mid b$

Left Linear Grammer

$A \to Bx$

$A \to x$

where $A, B \in V$, $x \in T$

eg $\to S bb \mid b$

— Simplify of Grammer —

1) Removal of the useless symbols
2) Removal of unit production
3) Removal of ϵ production

Removal of useless symbols →

$T \to aaB \mid abA \mid aaT$

$A \to aA$ → Also useless as A can't be terminated ✗

$B \to ab \mid b$

$C \to ad$ → T does not have C ∴ Therefore eliminating C ✗

- If a symbol can be useless if it does not appear on the right hand side of the production.
- If the symbol does not terminates.

$T \to aaB \mid aaT$

$B \to ab \mid b$

} new reduced grammer

Removal of Unit Production →

$\alpha \to \beta$

and, $\alpha, \beta \in V$

$S \to AB$

$A \to a$

$B \to C/b$

$C \to D$

$D \to E$

$E \to a$

$B \to C$

$C \to D$

$D \to E$

$B \to C \to D \to E$

$B \to E \to a$

$B \to a$

eliminating $C \to D$,

$D \to E$

as it follows $\alpha \to \beta$

$\alpha, \beta \in V$

and removing ϵ as not on the right side of the production rule.

$S \to AB$

$A \to a$

$B \to a/b$

$S \to AB$

$A \to a$

$B \to c/b$

$C \to a$

$S \to AB$

$A \to a$

$B \to a/b$

$S \to aa/ab$

Removal of Null moves $\longrightarrow$

$S \to ABC$

$A \to aA/\epsilon$

$B \to bB/\epsilon$

$C \to c$

Nullable —

$\{A, B\}$ identify the null variables

$S \to ABC/BC/AC/C$

$A \to a$

$B \to b$

$C \to c$

Replacing A, B with ϵ and eliminating the output

Chomsky Normal form

1) $A \to BC$

or

$A \to a$

where $\begin{cases} A, B, C \in V \\ a \in T \end{cases}$

$A \to \varepsilon$ (in class)

2) Number of steps required to generate a string of length 'n' is "$2|n|-1$"

$S \to AB$

$A \to a$

$B \to b$

S

$1 - AB$

$2 - aB$

$3 - ab$

$\begin{cases} AB = 2 \\ n = 2(2)-1 = 3 \end{cases}$

3) Used in Membership Algorithm

4) Passes tree obtained always binary tree

5) Length of each production is Restricted

Greibach Normal form

1) $A \to ax$

where $\begin{cases} x \in V^* \\ a \in T \end{cases}$

$A \to \varepsilon$

2) Number of steps required to generate a string of length 'n' is "n"

S

$1 - aB$

$2 - ab$

$\begin{cases} aB = 2 \\ n = 2 \end{cases}$

3) Used to convert CFG to PDA

Push Down Automata

4) Not always binary tree

5) Not Restricted

CFG → GNF →

$S \to CA \mid BB$

$B \to b \mid SB$

$C \to b$

$A \to a$

$S \to A_1$

$C \to A_2$

$A \to A_3$

$B \to A_4$

$A_1 \to A_2 A_3 \mid A_4 A_4$

$A_4 \to b \mid A_1 A_4$

$A_2 \to b$

$A_3 \to a$

$A_i < A_j$

$A_4 \to b \mid A_2 A_3 A_4 \mid A_4 A_4 A_4$

↓ — why we only converted $A_2 \to b$ and not $A_3 \to a$. is so we can achive Greibach NF.

$A_4 \to b \mid b A_3 A_4 \mid \underline{A_4 A_4 A_4}$

left recursion

⇒ aV*

↳ one terminal only.

$Z \to A_4 A_4 \mid A_4 A_4 Z$

→ $A_4 \to b \mid b A_3 A_4 \mid bZ \mid b A_3 A_4 Z$

$A_1 \to A_2 A_3 \mid A_4 A_4$

$A_4 \to b \mid b A_3 A_4 \mid bZ \mid b A_3 A_4 Z$

$Z \to A_4 A_4 \mid A_4 A_4 Z$

$A_2 \to b$

$A_3 \to a$

$A_1 \to bA_3 \mid b \mid bA_3 A_4 \mid bZ \mid bA_3 A_4 A_4 \mid bZA_4 \mid bA_3 A_4 Z A_4$

$A_4 \to b \mid bA_3 A_4 \mid bZ \mid bA_3 A_4 Z$

$Z \to bA_4 \mid bA_3 A_4 A_4 \mid bZA_4 \mid bA_3 A_4 Z A_4 \mid bA_4 Z \mid bA_3 A_4 A_4 Z \mid bZA_4 Z \mid bA_3 A_4 Z A_4 Z$

$A_2 \to b$

$A_3 \to a$

$S \to CA \mid BB$

$B \to b \mid SB$

$C \to b$

$A \to a$

$S \to bA \mid BB$ $\quad\quad$ $S \to A_1$

$B \to b \mid SB$ $\quad\quad$ $A \to A_2$

$A \to a$ $\quad\quad$ $B \to A_3$

$A_1 \to bA_2 \mid A_3A_3$

$A_3 \to b \mid A_1A_3$

$A_2 \to a$

$\to A_3 \to b \mid bA_2A_3 \mid A_3A_3A_3$

$Z \to A_3A_3 \mid A_3A_3Z$

$A_3 \to b \mid bA_2A_3 \mid bZ \mid bA_2A_3Z$

$A_2 \to a$

$A_1 \to bA_2 \mid bA_3 \mid bA_2A_3A_3 \mid b_2A_3 \mid bA_2A_3ZA_3$

$Z \to bA_3 \mid bA_2A_3A_3 \mid b_2A_3 \mid bA_2A_3ZA_3$

$bA_3Z \mid bA_2A_3A_3Z \mid b_2A_3A_3Z \mid bA_2A_3ZA_3$

$S \to a \mid aA \mid B$

$A \to aBB \mid \varepsilon$ $\quad\quad$ $\{A\}$

$B \to Aa \mid b$

$S \to a \mid aA \mid B$

$A \to aBB$

$B \to Aa \mid b$

$S \to a \mid aA \mid Aa \mid b$

$A \to aBB$

$B \to Aa \mid b$

$S \to XA \mid BB$

$B \to b \mid SB$

$X \to b$

$A \to a$

$A_1 \to A_2A_3 \mid A_4A_4$

$A_4 \to b \mid A_1A_4$

$A_2 \to b$

$A_3 \to a$

$S \to A_1$

$X \to A_2$

$A \to A_3$

$B \to A_4$

CNF →

$A_1 \quad A_1\ A_2\ A_3 \quad A_4$

$S \to XA \mid BB$

$B \to b \mid SB$

$X \to b$

$A \to a$

$A_1 \to A_2A_3 \mid A_4A_4$

$A_4 \to b \mid A_1A_4$

$A_2 \to b$

$A_3 \to a$

$A_1 \to bA_3 \mid A_4A_4$

$A_4 \to b \mid bA_3A_4 \mid A_4A_4A$

$A_4 \to b \mid bA_3A_4 \mid b2 \mid bA_3A_42$

$2 \to A_4A_42 \mid A_4A_4$

$A_1 \to bA_3 \mid bA_4 \mid bA_3A_4A_4 \mid b2A_4 \mid bA_3A_4A_42 \mid bA_42A_4$

$2 \to bA_42 \mid bA_3A_4A_42 \mid b2A_42 \mid bA_3A_42A_42$

$A_2 \to b$

$A_3 \to a$

$A_4 \to b \mid bA_3A_4 \mid b2 \mid bA_3A_42$

$A_2 \to b$

$A_3 \to a$

Now Replace $A_4 \to$ with $A_4 \to \ldots$.

$S \to bA \mid aB$

$A \to bAA \mid aS \mid a$

$B \to aBB \mid bS \mid b$

$X \to b$

$S \to XA \mid aB$

$A \to XAA \mid aS \mid a$

$B \to aBB \mid XS \mid b$

$Y \to AA$

$C \to BB$

$S \to XA \mid aB$

$A \to XY \mid aB \mid a$

$B \to aC \mid XS \mid b$

$Y \to AA$

$C \to BB$

$D \to a$

$S \to XA \mid DB$

$A \to XY \mid DB \mid a$

$B \to DC \mid XS \mid b$

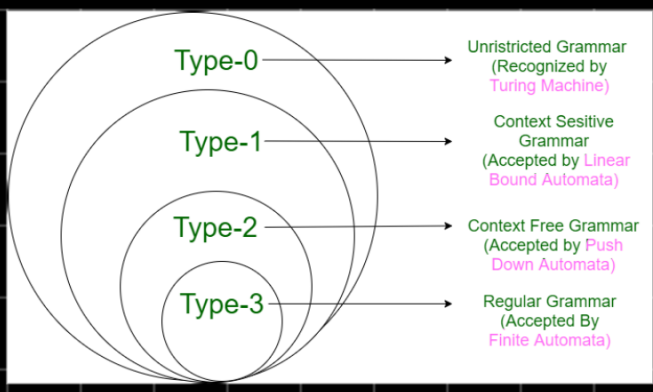
$Y \to AA$

$C \to BB$

$D \to a$

$X \to b$

Chomsky Hierarchy → Can not be null production



1. Type '0' → Turing Machine

   $A \to \beta$ — can be anything including '$\in$'

   At least one variable on the left hand side.

   $\beta \in (\Sigma \cup V_n)^*$ V — Variable.

   $A \in (\Sigma \cup V_n)^* V_n (\Sigma \cup V_n)^*$

2. Type '1' → Linear Bounded Automata

   $\alpha \to \beta$

   At least one Variable on the left hand side.

   $\alpha \leq \beta$ length

   β — (set of symbols and variables)

   - $\beta \in (\Sigma \cup V_n)^+$
   - $S \to \in$ / S can't appear on the right hand side.

3. Type '2' → Push Down Automata

   $\alpha \to \beta$

   $\alpha \in V_n$, length = 1

   $\beta \in$ (set of variables)*

4. Type '3' → Finite Automata

   Regular Grammer → Right / Left linear grammer.

Push down Automata → "PDA"

No Transition Table, due to complexity

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

$Q, \Gamma, \Sigma, F, \delta, Z, \Gamma$ Δ, f

$Q \times (\Sigma \cup (\in)) \times$

- Q — finite set of states (like FA)
- Σ — Input symbol
- Γ — Static Alphabet

Γ ⇒ Alphabet

Z_0 ⇒ Stack $Q \times (\Sigma \cup \in) \times \Gamma$

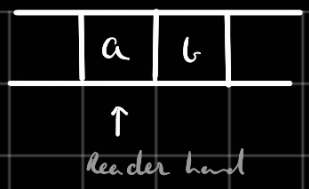
- δ — Transition function

  $\delta : Q \times (\Sigma \cup \in) \times \Gamma$

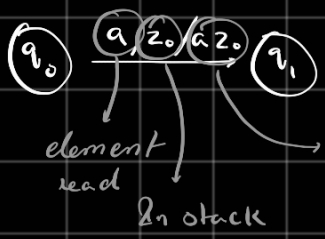
- q_0 — Initial State
- Z_0 — Stack start symbol
- F — Final State

Γ → take $Q \times \Sigma \to Q \Gamma^{*}$

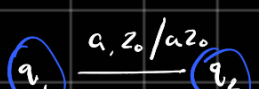
β →



at 'q_0' to 'q_1' we read 'a' and 'z_0' the element in the stack.

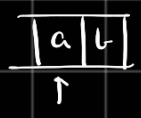


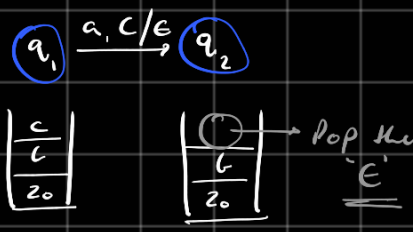
1) Push



$\delta(q_1, a, z_0) = \delta(q_2, az_0)$

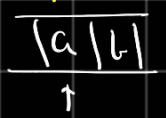
2) Pop

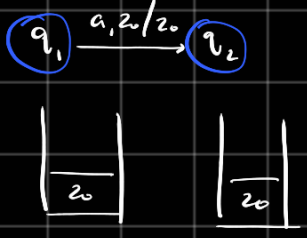




$\delta(q_1, a, c) = \delta(q_2, \epsilon)$

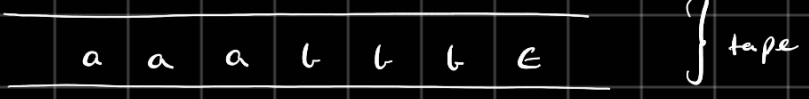
3) Skip

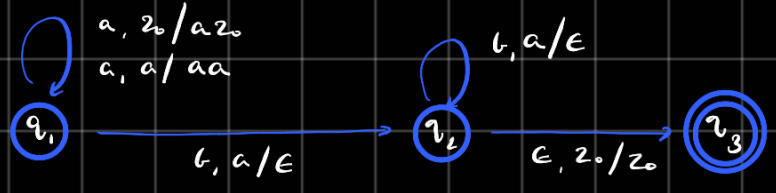


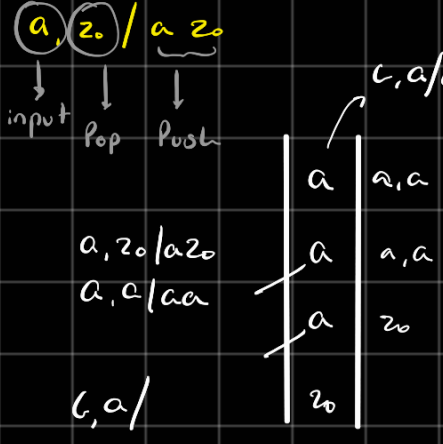


$\delta(q_1, a, z_0) = \delta(q_2, z_0)$

Example — $L = \{a^n b^n \mid n \geq 1\}$







$\delta(q_1, a, z_0) \rightarrow \delta(q_1, az_0)$

$\delta(q_1, a, a) \rightarrow \delta(q_1, aa)$

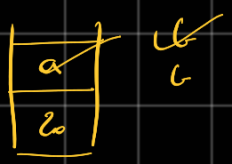
$\delta(q_1, b, a) \rightarrow \delta(q_2, \epsilon)$

$\delta(q_2, b, a) \rightarrow \delta(q_2, \epsilon)$

$\delta(q_2, \epsilon, z_0) \rightarrow \delta(q_3, \epsilon)$ // $\delta(q_2, \epsilon, z_0) \rightarrow \delta(q_3, z_0)$

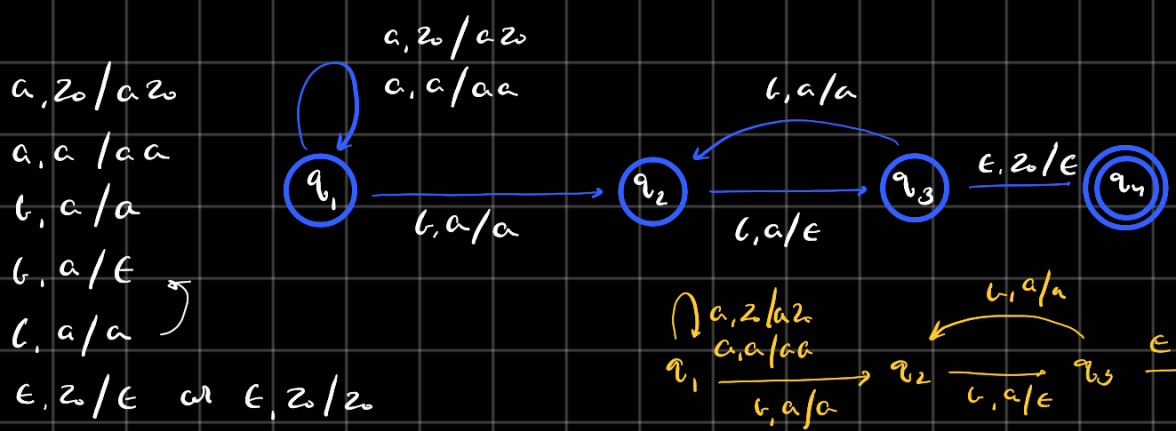
Q — $L = \{a^n b^{2n} / n \geq 1\}$

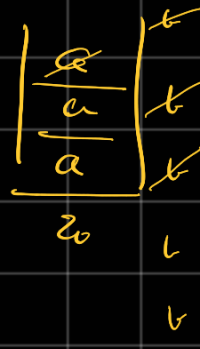
a b b



a	a	a	b	b	b	b	b	b	∈

↑





Q — $L = \{w c w^R \mid w \in (a,b)^*\}$

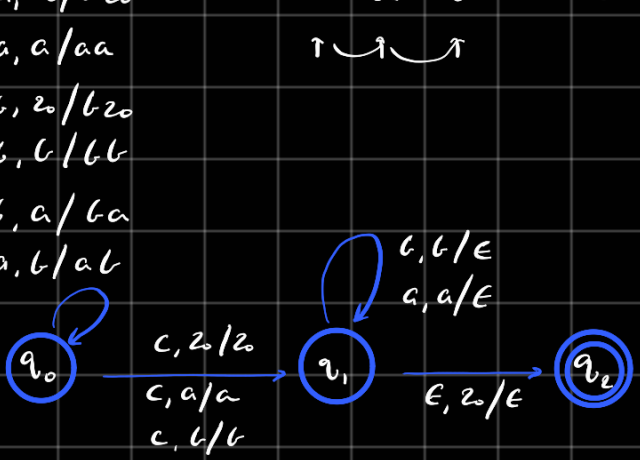
a b a c a b a

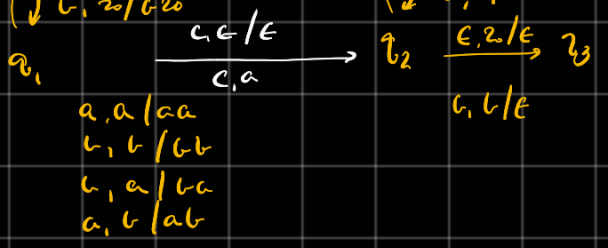
b a b c b a b

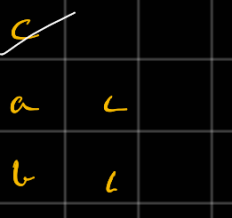
c

w — a b c

w^R — c b a







wcw

not possible

$w = abb$

c

w^R won't be able to access 'a' as its below in the stack.

Turning Machine →

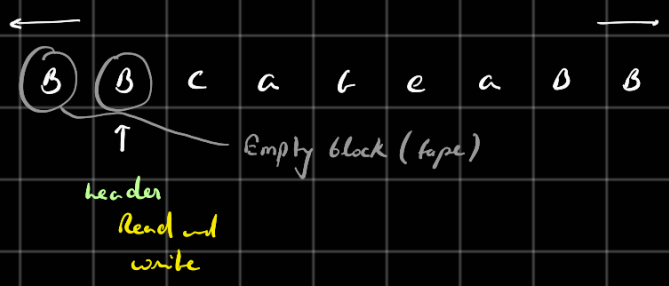
$TM = \{\Sigma, Q, q_0, F, \Gamma, B, \delta\}$

Both side infinite streach

- Q — set of states
- Σ — set of symbols
- q_0 — set of initial states
- F — set of final states
- Γ — tape symbol
- δ — transition function

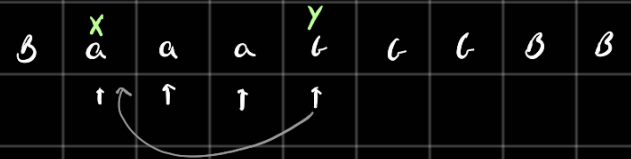
$\delta : Q \times \Sigma \longrightarrow Q \times \Gamma \times (L/R)$

- B — blank symbol

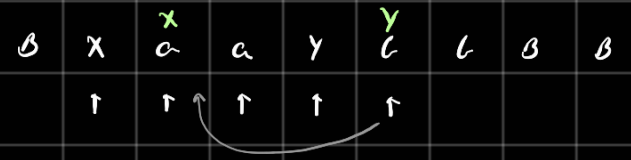


Push down automata + 2 stacks = Turning machine

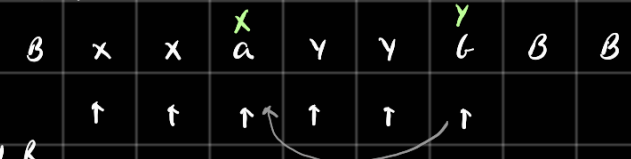
$Q \rightarrow L = \{a^n b^n / n \geq 1\}$



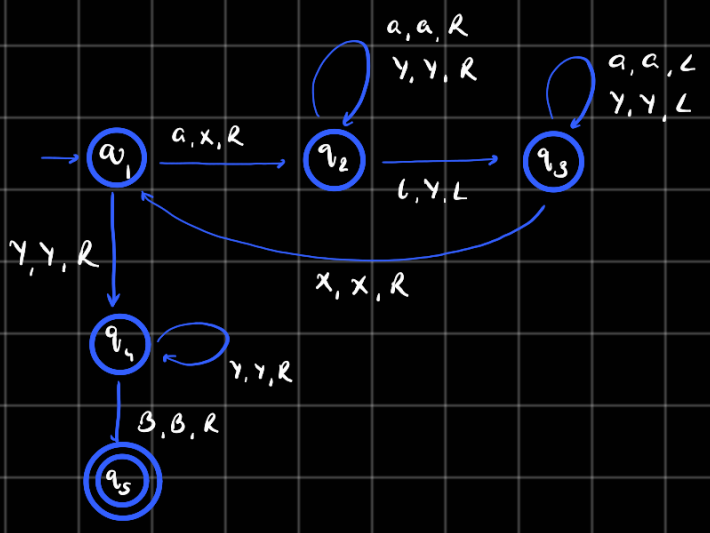
a, X, R
a, a, R
b, Y, L

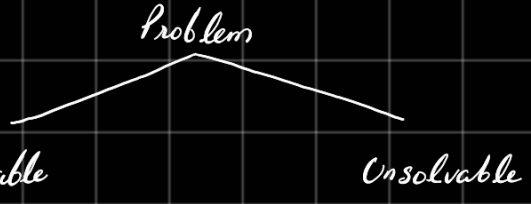


X, X, R
a, X, R
a, a, R
Y, Y, R
b, Y, L



Y, Y, R
B, B, R





- If their exists an potential solution.
- If proved mathamatical that their exists no possible solution.

If no potential solution exists for the problem.

Decidability and Undecidability

Recursive Language → **Decidable**

Turing machine will always halt either accept / reject and halt.

Recursively Enumerable Language → **Partialy Decidable**

Turing machine will halt only when it is excepting the string in the language.

In other cases it may or may not halt.

Decidable Language:
A language 'L' is decidable if it is a recursive language. All decidable languages are recursive languages and vice-versa.

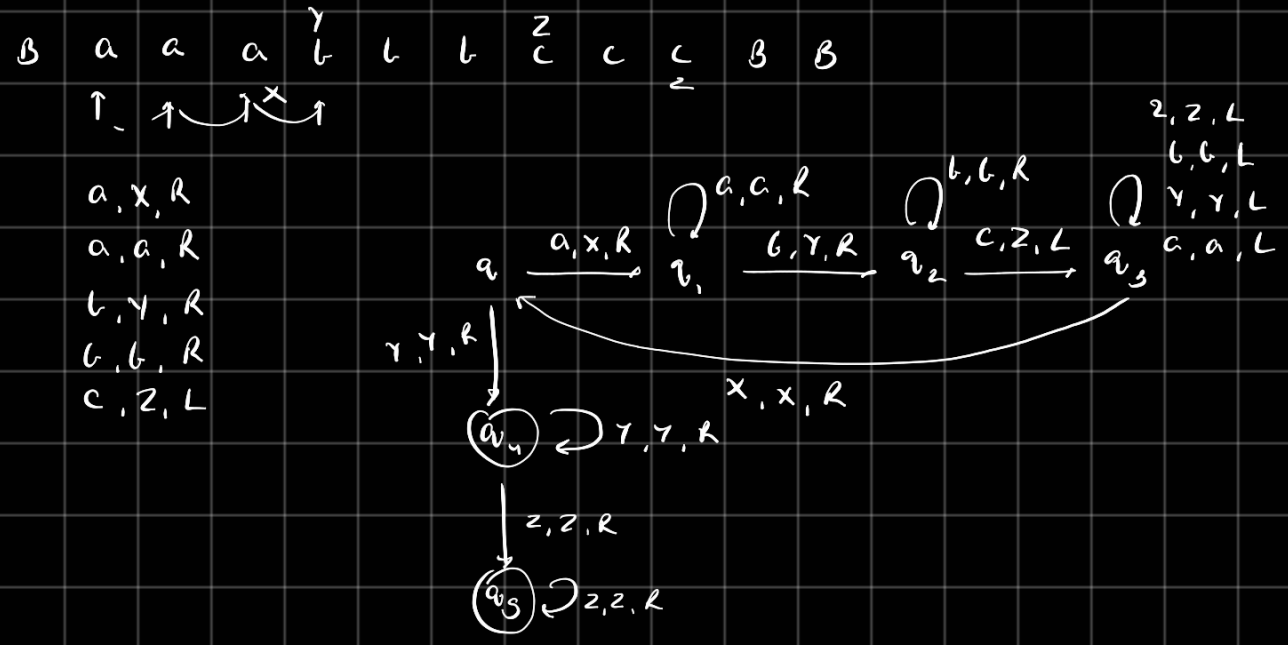
Partially Decidable Language:
A language 'L' is partially decidable if 'L' is a recursively enumerable language.

Undecidable Language:

- A language is undecidable if it is not decidable.
- An undecidable language may sometimes be partially decidable but not decidable.
- If a language is not even partially decidable, then there exists no Turing machine for that language

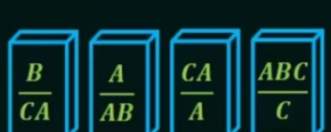
Recursive Language	TM will always Halt
Recursively Enumerable Language	TM will halt sometimes & may not halt sometimes
Decidable Language	Recursive Language
Partially Decidable Language	Recursively Enumerable Language
UNDECIDABLE	No TM for that language

$a^n b^n c^n$

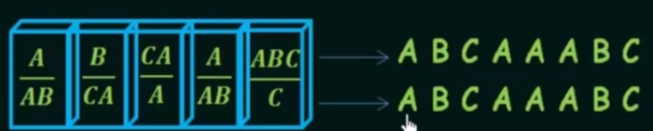


Post correspondence Problem → PCP

Dominos:



We need to find a sequence of dominos such that the top and bottom strings are the same.



eg :-

	A	B
①	1	111
②	10111	10
③	10	0

② ① ① ③

A : 10111101

B : 10111110

Modified Post Correspondence Problem(MPCP

Let A and B be two non-empty list of strings over Σ. A and B are given as below:

$A = \{w_1, w_2, w_3 \ldots w_k\}$

$B = \{v_1, v_2, v_3 \ldots v_k\}$

Then we can say ,there is a modified post correspondence between A and B is there is a sequence of one or more integers i,j,k...m such that :

The string $w_1, w_i, w_j, w_k \ldots w_m$ is equal to $v_1, v_i, v_j, v_k \ldots v_m$.

first is fixed

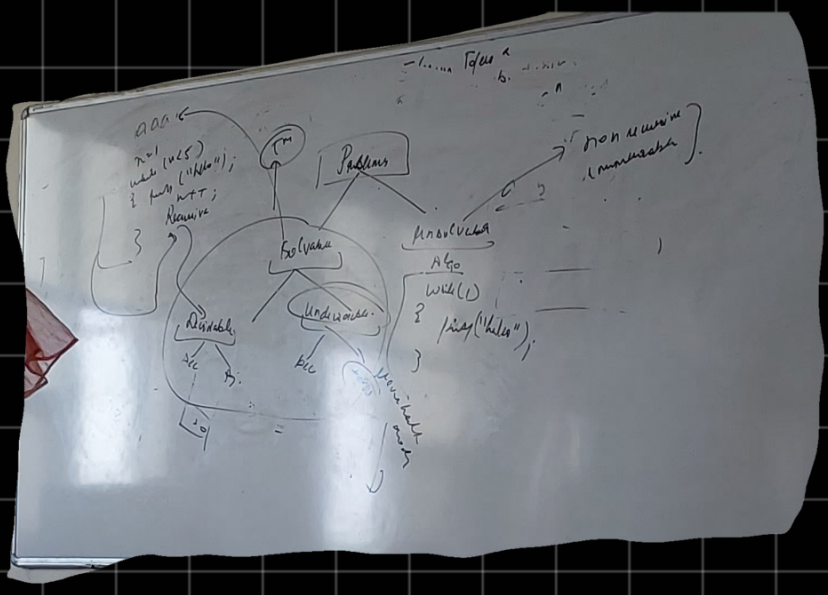
in PCP $w_i, w_j, w_k \ldots w_m$ $v_i, v_j, v_k \ldots v_m$

no value is first fixed

PCP (Post correspondence Problem)

undecidable problems.

MPCP



0 0 0 0 1 1

0,20/020
0,0/00

$q_0 \longrightarrow q_1$

1, 0/ϵ

1 1 ∞ 0 0 0