Introduction

- **Computation:** Process of solving problems using a sequence of steps
- Processing input → producing output
- Can be performed by:
  - Human (manual computation)
  - Machine (computer computation)
  - Mathematical models (automata, Turing Machine)

- **Symbol:** smallest unit of information
  e.g: a, b, 0, 1 // A single digit or letter

- **Alphabet:** A finite set of symbols
  Represented as Σ
  e.g $\Sigma = \{0,1\}$ — Binary Alphabet
  $\Sigma = \{a,b,c\}$ — English letters subset

- **String:** A finite sequence of symbols taken from an alphabet // concatenation of finite symbols from alphabet
  e.g: $\Sigma = \{0,1\}$ → '101', '1100'
  $\Sigma = \{a,b\}$ → 'abb', 'aaa'

- **Substring:** A portion of a string
  e.g string = automata
  Substring = auto, tom, ta, mata

- Two Types of Languages
  1. Formal (Syntactic)
  2. Informal (Semantic)

- **Length of string $|s|$:**
  - $s = 10101 \rightarrow |s| = 5$
  - $s = aba \rightarrow |s| = 3$
  - Empty string (ε) has length 0

- **Language:** set of strings formed from an alphabet
  - $\Sigma = \{0,1\}$
  - $L = \{00, 1, 01, 10, 0011\}$

  - Infinite languages are possible

- **Words:** strings belonging to some language
  - $\Sigma = \{x\}$
  - $L = \{x^n : n = 1,2,3 \dots\}$ or $L = \{x, xx, xxx \dots\}$. $x, xx \dots$ are words of L
  - All words are strings, but not all strings are words

Operations on Strings

- **Power of Strings:**
- $w =$ "ab"

repeat the whole string

- $w^0 = \varepsilon$
  $w^1 = ab$
  $w^2 = abab$
  $w^3 = ababab$
- non-negative powers
- **Kleene-Star:** set of all strings formed by concatenating 0 or more strings from the alphabet
- $A^* = \bigcup_{i=0}^{\infty} A^i = A^0 \cup A^1 \cup A^2 \ldots$
- $\Sigma =$ Alphabet

  $\Sigma^* =$ set of all possible substrings (including ε)
- $\Sigma = \{a, b\}$

  $\Sigma^* = \{\varepsilon, a, b, ab, aab \ldots\}$
- Any language L over Σ is a subset of Σ^*
  $L \subseteq \Sigma^*$
- Each subset of Σ formed by powers is unique
- $\Sigma^0 = \varepsilon$ $\quad\quad\quad\quad\quad\quad \Sigma = \{a\}$
  $\Sigma^1 = \{a\}$
  $\Sigma^2 = \{a, a\}$
- Each Σ^n is distinct
- **Kleene Plus:** $A^+ = A^* - \{\varepsilon\}$

Concatenation of Strings

- Assosciative: $(xy)z = x(yz)$
- Identity: $x\varepsilon = \varepsilon x = x$
- Not commutative: $xy \neq yx$ in general

  e.g "ab" · "c" = "abc" $\neq$ "cab"

- Length Property: $|xy| = |x| + |y| = |yx|$

  e.g $x =$ "ab", $y =$ "cd"

  $|xy| = |$"abcd"$| = 4$

  $|yx| = |$"cdab"$| = 4$

- Reverse of a String: Rev(s) // s^r

- Double-Reverse Property: $(w^R)^R = w$
- Length Property: $|w^R| = |w|$
- Concatenation Property: $(xy)^R = y^R x^R$

  e.g $x =$ "ab", $y =$ "cd"

  $(xy)^R = ($"abcd"$)^R =$ "dcba"

  $y^R x^R =$ "dc" · "ba" = "dcba"

- Unary Alphabet Property: $\Sigma = \{a\}$, $w = a^n \rightarrow w^R = w$

- Palindrome:
- If $w = U.U^R \rightarrow$ even palindrome for $U \in \Sigma^*$

- String is split in 2 equal halves, and the second half is the reverse of the first

- $U =$ "ab" $\rightarrow w = U.U^R =$ "ab" . "ba" = "abba"

- If $w = U.x.U^R$, for some $U \in \Sigma^*$ and $x \in \Sigma$ $\rightarrow$ odd palindrome
- Here the string has a middle character x, and parts after and before it are mirror images

- $w = U.x.U^R =$ "ab"."c"."ba" = "abcba"

- Prefix
- Reading string from the front side

- string= "abcde"

- Prefixes:
- a
- ab
- abc
- abcd
- adcde

suffix, prefix, lexical analy

- Suffix
- Reading from the end

- string= "abcde"

- Suffixes:
- "e"
- "de"
- "cde"
- "bcde"
- "abcde"

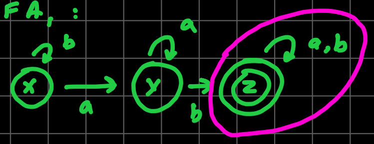
- $\text{Prefix}(u) = \{ x \mid u = xy,\ y \in \Sigma^* \}$

- $\text{Suffix}(u) = \{ x \mid u = xy,\ x \in \Sigma^* \}$

zer.

Concatenation

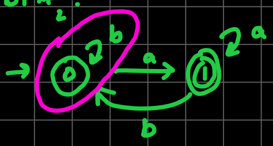
- DFA$_1$:



Q- $L_1 . L_2$ → final, initial

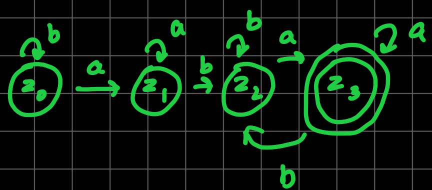
$L_2 . L_1$ → final, initial

- DFA$_2$:



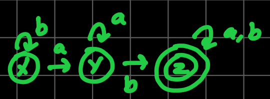
· Current State	a	b
- z_0 (x)	y	x
- z_1 (y)	y	(z,0)
- z_2 (z,0)	(z,0,1)	(z,0)
- z_3 (z,0,1)	(z,0,1)	(z,0)

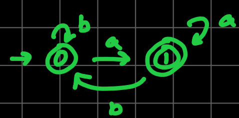
→ stop if no new final state



Union & Intersection

- Union: X+Y
- Intersection: X.Y



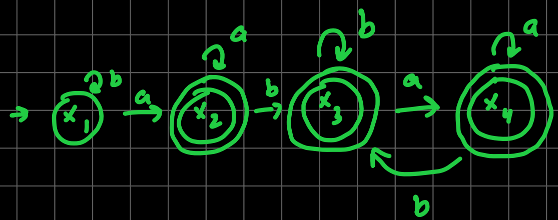


Start from initial of both

- Union

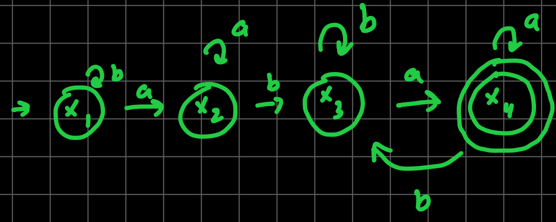
- = initial
+ = final

Current state	a	b
- x_1 (x,0)	(y,1)	(x,0)
+ x_2 (y,1)	(y,1)	(z,0)
+ x_3 (z,0)	(z,1)	(z,0)
+ x_4 (z,1)	(z,1)	(z,0)



- Intersection

Current state	a	b
$- \; x_1 \; (x,0)$	$(y,1)$	$(x,0)$
$- \; x_2 \; (y,1)$	$(y,1)$	$(z,0)$
$- \; x_3 \; (z,0)$	$(z,1)$	$(z,0)$
$+ \; x_4 \; (z,1)$	$(z,1)$	$(z,0)$



make this final state when both DFA's final state combine, diagram and table are same for both

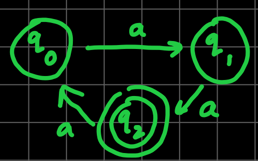
Transition Function

- $\delta : Q \times \Sigma \to Q$
- $\hat{\delta} : Q \times \Sigma^* \to Q$] DFA

string

$\delta : Q \times \Sigma \to 2^Q$

$\hat{\delta} : Q \times \Sigma^* \to 2^Q$] NFA



- aaa accepts?

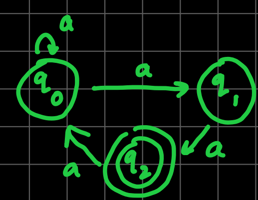
$\hat{\delta}(q_0, aaa)$

$= \delta(\hat{\delta}(q_0, aa), a)$

$= \delta(\delta(\hat{\delta}(q_0, a), a), a)$

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

$= \delta(q_2, a) = \underline{\underline{q_0}}$ not final state, so does not accept



aaa?

$\hat{\delta}(q_0, aaa)$

$= \delta(\hat{\delta}(q_0, aa), a)$

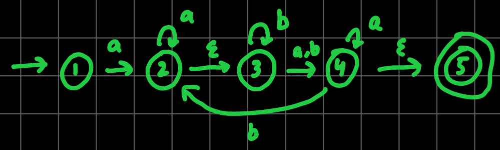
$= \delta(\delta(\hat{\delta}(q_0, a), a), a)$

$= \delta(\delta(q_0, q_1, a), a)$

$= \delta[(\delta(q_0, a) \cup \delta(q_1, a)), a]$

$= \delta(q_0, q_1, q_2, a)$

$= \underline{\underline{q_0, q_1, q_2}}$ = final state present so accepts



	a	b
1	2,3	2,5,6,7,1
2	3	5,6,7,1,2
3	-	4,1,2,5,6,7
4	4,1,2,3	1,2,5,6,7
5	-	6,7,1,2,5
6	5	-
7	1,2,3	1,2,5,6,7

Epsilon Closure - 1

$\hat{\delta}(1,\varepsilon) = 1$

$\hat{\delta}(2,\varepsilon) = 2,3$

$\hat{\delta}(3,\varepsilon) = 3$

$\hat{\delta}(4,\varepsilon) = 4,5$

$\hat{\delta}(5,\varepsilon) = 5$

2

	a	b
1	2,3	-
2	2,3,4	-
3	4,5	3,4,5
4	4,5	2
5	-	-

All accessible states

$\hat{\delta}(1,ab)$

$= \delta(\hat{\delta}(1,a),b)$

$= \delta(2,3,b)$

$= \delta(2,b) \cup \delta(3,b)$

$= \{2,3,4\} \cup \{3,4,5\}$

$= \{2,3,4,5\}$

$ab \in L$, 5 is included

$\hat{\delta}(1,abb)$

$\delta(\delta(\hat{\delta}(1,a),b),b)$

$= \delta(\delta(2,3,b),b)$

$= \delta(5,6,7,1,2,4,b)$

$= 6,7,1,2,5$

accept

Non-Regular Lang

- Infinite language
- Infinite State

Regular lang → Finite state

- Pumping Lemma (-ve test):

Man needs 2 cookies for survival
- True
  - may survive
  - may not survive
- False
  - confirm dead

-ve test

Man needs 2 cookies for survival
- True
  - confirm survive
- False
  - may dead
  - may not dead

+ve test

1. Suppose language is regular
2. If regular, then there must be a FA with n states
3. word should belong to L & |w| >= n
4. If 3 true, then there must be a loop
5. w= xyz

w= original string
x= Before y
y= content of first loop
z= Remaining string

CYK Algo

- Checks whether the string belongs to a given CFG
- only works when grammer is in CNF

Q- S → AB | BC
A → BB | a
B → BA | b
C → AC | AA | a

w= ababb
12345

- use direct production rule

	5	4	3	2	1
1	S	C	S	S	A,C
2	S,B	A	B,S	B	
3	C	S	A,C		
4	A	B			
5	B				

(12)
11 . 22
(A,C) . (B)
AB , CB
S x

(2 3)
(22) (33)
B . (A,C)
BA , BC
B , S

3 4
33 44
(A,C) . B
AB , CB
S , x

4 5
44 55
B . B
BB
A

1|2 34

1 4
11 24
A,C A
AA , CA
C x

12 34
S S
SS
x

13 44
S B
SB
x

1|23

1 3
11 23
(A,C) . (B,S)
AB , AS , CB , CS
S x x x

12|3
12 23
S . (B,S)
SB . SS
x x

2 5
22 35
B C
BC
S

23 45
(B,S) A
BA , SA
B x

24 55

2 4
22 34
B S
BS
x

23 44
(B,S) (B)
BB , SB
A x

3 5
33 45
(AC) A
AA CA
C x

34 55
S B
SB
x

1 , 5
11 25
A,C S,B
AS AB
x S

12 35
S C
x

13 45
S A
x

14 55
C B
x

CS CB
x x

Q- S → AB | BB
A → BA | c | b) CC
B → BB | a | b | c
C → AA

abccba
123456

	6	5	4	3	2	1
1	S,B	ACSB	A,C,S,B	,C,S,B	A,S,B	B
2	S,B	CSAB	A,B,C,S	C,S,A,B	A,B	
3	S,B	C,S,A,B	C,S,A,B	A,B		
4	S,B	C,S,A,B	A,B			
5	S,B	A,B				
6	B					

1 6
11 26 12 36 13 46 14 56 15 66
B SB
BB
SB

12
11 22
B (A,B)
BA BB
A S,B

2 3
22 33
(A,B) (A,B)
AA , AB , BA, BB
C S A B

3 4
33 44
(A,B) (A,B)

45
44 55
(A,B) (A,B)

56
55 66
(A,B) (B)
AB BB
S B

1 3
11 23
B (C,S,A,B)
BC, BS, BA, BB
X X A S,B

12 33
(A,S,B) (A,B)
AA, AB, ~~SA~~, ~~SB~~, ~~BA~~, BB
C S

(2 , 4)
22 34
(A,B) (C,S,A,B)
~~AC~~, ~~AS~~, AA, AB,
C S
~~BC~~, ~~BS~~, BA, BB
A S, B

23 44
(C,S,A,B) (A,B)
~~CA~~ , ~~CB~~ , ~~SA~~, ~~SB~~
AA, AB , ~~BA~~, ~~BB~~
C S

3 5
33 45
(A,B) (C,S,A,B)
~~AC~~, ~~AS~~, AA, AB
C S
~~BC~~, ~~BS~~, BA, BB
A S, B

34 55
(C,S,A,B) A,B
~~CA~~ , ~~CB~~ ~~SA~~, ~~SB~~
~~AA~~, ~~AB~~ , ~~BA~~, ~~BB~~

4, 6
44 56
(A,B) (S,B)
~~AS~~ AB
S
~~BS~~ BB
B

45 66
(C,S,A,B) (B)
~~CB~~, ~~SB~~, ~~AB~~, ~~BB~~

(1,4)
11 24
B (A,B,C,S)
BA BB ~~BC~~ ~~BS~~
A S,B

12 34
ASB CSAB
~~AC~~ ~~AS~~ AA AB
C S
~~BC~~ ~~BS~~ ~~BA~~ ~~BB~~

13 44
A,C,S,B (A,B)
~~AA~~ ~~AB~~ ~~CA~~ ~~CB~~
~~BA~~ ~~BB~~

2 5
22 35
(A,B) (C,S,A,B)
~~AC~~ ~~AS~~ AA AB
C S
~~BC~~ ~~BS~~ BA BB
A S,B

23 45
C,S,A,B CSAB
CC ~~CS~~ CA ~~CB~~
A
AA AB BA BB
C S A S,B

24 55
(A,B,C,S) (A,B)
AA AB BA BB
C S A B
~~CA~~ ~~CB~~

3 6

33 46

A,B S,B

AB , BB

S S,B

34 56

(C,S,A,B) (S,B)

~~AB~~, ~~BB~~

35 66

(C,S,A,B) B

~~AB~~, ~~BB~~

15

11 25

B CSAB

BB , BA

S,B

12 35

ASB CSAB

AA , AB ,

C S

13 45

CSB CSAB

CC

A

14

ACSB

55

A,B

2 6

22 36

A,B S,B

AB BB

S S,B

23 46

CSAB S,B

24 56

ABCS SB

25 66

CSAB B

Turing Machine

- Abstract Machine
- → Having infinite tape
- → Read/write head
- → Finite set of states
- → Transition function

TM
- Acceptor
  - FA
  - PDA
  - LBA
- Transducer

- 7 tuple model

$M = (Q, \Sigma, \Gamma, \delta, blank, q_0, q_f)$

Q = set of states
Σ = Input Alphabet
Γ = Tape Alphabet
δ = Transition Function
blank = Blank Symbol
q_0 = start state
q_f = final/accept state

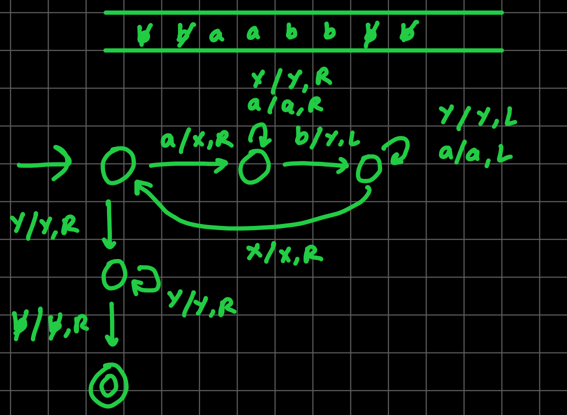
$\delta : \underbrace{Q\,\Gamma}_{\text{input}} \rightarrow \underbrace{Q\,\Gamma\,D}_{\text{output}}$

- There are blanks at the start and at the end of tape. when starting we assume we start from the first character. when string has ended, halt the machine using the blank character on the tape.

- Examples of TM as an acceptor

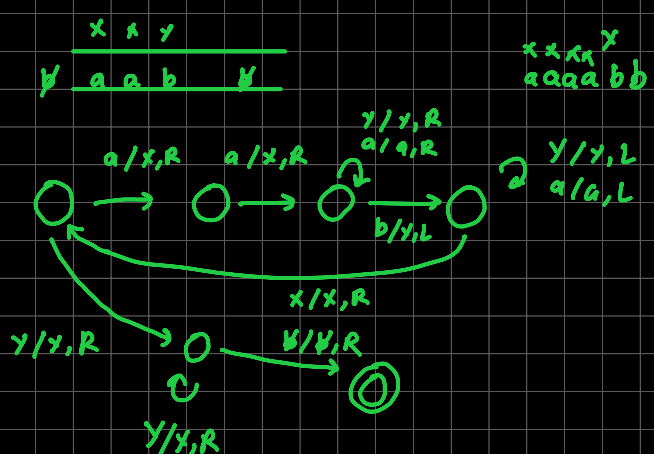
Q. $L = \{ a^n b^n \mid n \geq 1 \}$ aabb

$\cancel{b} = \Delta$



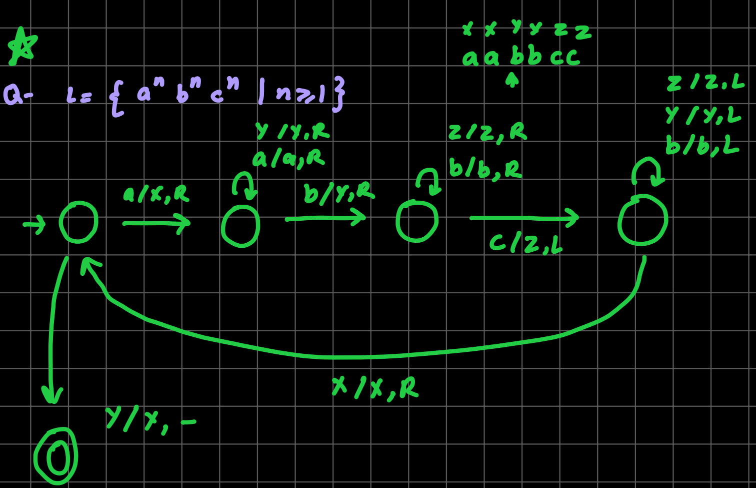
Q- $L = \{ a^{2n} b^n \mid n \geq 1 \}$

aab



☆

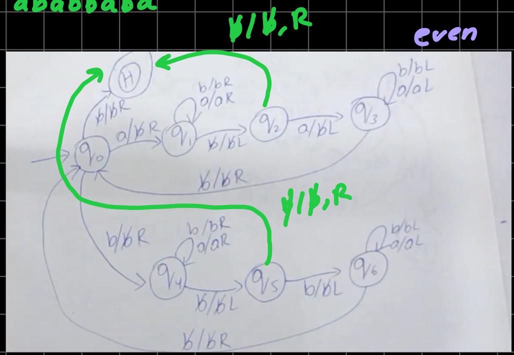
Q- $L = \{ a^n b^n c^n \mid n \geq 1 \}$



Q- Even Palindrome / odd palindrome

ababbaba

even



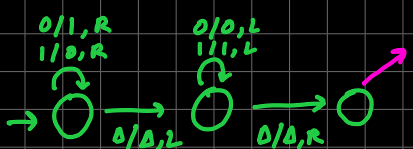
Q- na (w) = nb(w)

Transducer

· Operations

Q- 1's compliment

1 0 1 0 1 0



Output state
(must mention)

Q- 2's compliment

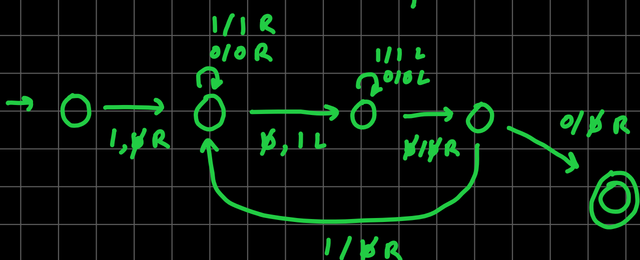
Sum

· Numbers are given in unary representation
· Numbers are seperated by a seperator

e.g: 2 + 6 = 8

1 1 0
1 1 0
0 0 1

seperator

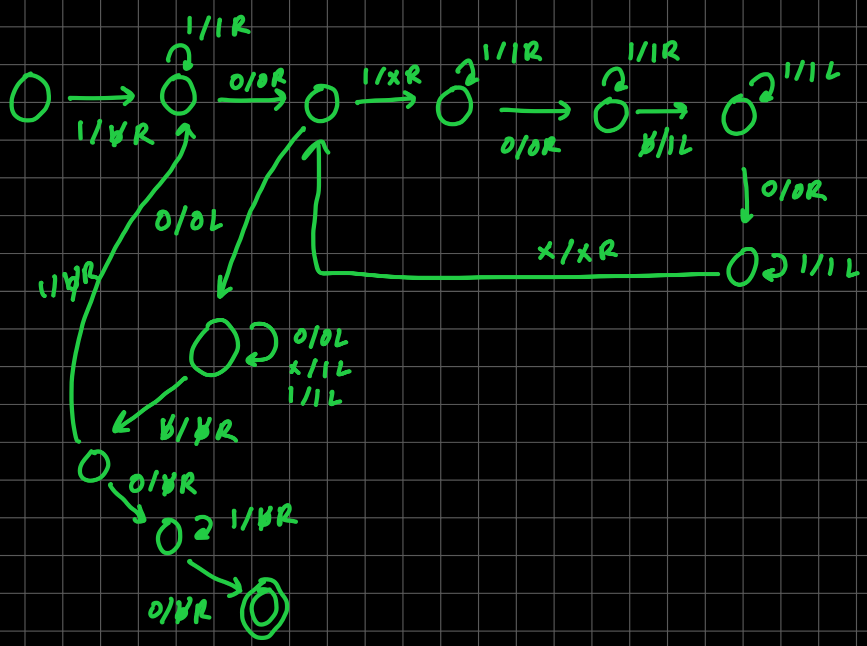


Multiply

- 2x3 = 2+2+2 = 6
- 3x2 = 3+3 = 6

seperator

ƀ ƀ 1 1 0 1 1 1 0 ƀ ƀ ƀ

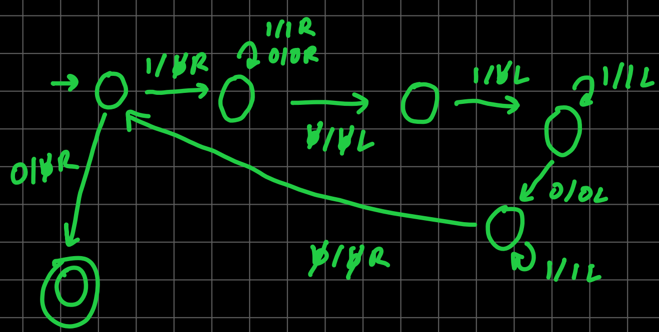


subtraction

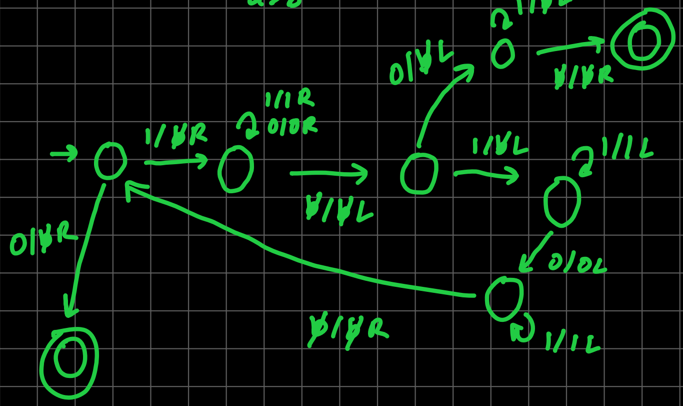
- 3 cases
- a < b
  3-2 = 1
- a = b
  2-2 = 0
- a > b
  2-3 = 0

a<b , a=b

ƀ 1 1 0 1 1 1 1 ƀ ƀ



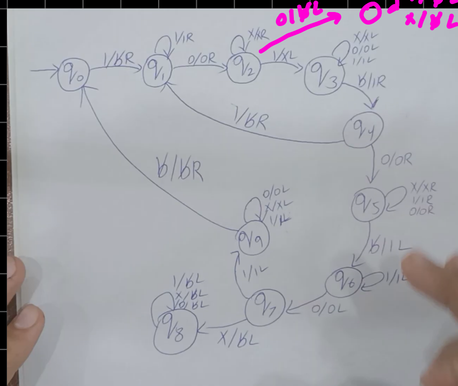
a>b



Division

- 6 ÷ 3 = $\underbrace{6 - (3+3)}_{\to\, 2} = 0$
- if answer in floating point, set ƀ on tape

0/ƀL 1/ƀL x/ƀL] for decimal



complete divison