Regular Expressions

① . Any terminal symbol i.e symbol $\in \Sigma$: $\frac{a, b, c}{\text{Terminal symbol}}$

② The union of two regular expressions is also a regular expression . R_1, R the $(R_1 + R_2)$

Λ — Empty, ϕ — Null

③ The concatenation of two regular expression also a regular expression . r_1, r_2 $(R_1 . R_2)$

④ The iteration (or closure) of a regular expression also a reg ex . $R \rightarrow R^*$, $a^* = \Lambda, a, aa, aaa, aaaa,$

⑤ The regular expression over Σ are precisely those obtained recursively by the application of the above rules once or several times

Describe the set as a regular expression:

① $\{0, 1, 2\}$ $R = 0 + 1 + 2$ [or symbol (+)]

2) $\{\Lambda, ab\}$ $R = \Lambda ab$ যেহেতু মধ্যে একটি symbol এবং শুধু Λ symbol থাকলে তখন + না দিলেও হবে,

③ $\{abb, a, b, bba\}$ $R = abb + a + b + bba$.

④ $\{\Lambda, 0, 00, 000, \ldots\}$ $R = 0^*$

⑤ $\{1, 11, 111, 1111, \ldots\}$ $R = 1^+$

Identities of regular expression.

(1) $\phi + R = R$ → Empty bot

(2) $\phi R + R\phi = \phi$

(3) $\epsilon R = R\epsilon = R$

(4) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

(5) $R + R = R$

(6) $R^* R^* = R^*$

(7) $RR^* = R^*R$

(8) $(R^*)^* = R^*$

(9) $\epsilon + RR^* = \epsilon + R^*R = R^*$

(10) $(PQ)^*P = P(QP)^*$

(11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

(12) $(P+Q)R = PR + QR$ and

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

Design Regular Expression for the following language over $\{a, b\}$

(1) Language accepting string of length exactly 2.

(11) n n n n n atleast 2

(3) n n n n n almost 2

$L_1 = \{aa, ab, ba, bb\}$

$R = aa + ab + ba + bb$

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

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

$L_2 = \{aa, ab, ba, bb, aaa, \dots\}$

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

$(a+b)(a+b)$ → দুটো হলো সহজ Atleast দুটো থাকতেই হবে

$(a+b)^*$ → এই দুইয়ের বেশি হলেও চলবে,

$L_3 = \{\epsilon, a, b, aa, ab, ba, bb\}$

$R = \epsilon + a + b + aa + ab + ba + bb$

$= (\epsilon + a + b)(\epsilon + a + b)$

→ A minor result whose sole purpose is to help in proving a theorem.

Pumping lemma (For Regular Expression)

~~Theorem 10~~

Pumping lemma ব্যবহার করা হয় কোনো language যে Reg Regular না তা প্রমাণ করার জন্য।

Regular language ~~that~~ are the language that can be design using ~~an~~ Finite state machine.

If A is a regular language then A has a pumping length 'p' such that any string 's' where length of $|s| \geq p$ may be ~~delivered~~ divided into 3 parts $s = xyz$ such that the following condition must be true.

1. $xy^iz \in A$ for every $i \geq 0$
2. $|y| > 0$
3. $|xy| \leq p$

- যেখানে Prove করতে বলছি, সেখানে Neg করে বলবা, যে Neg prove করতে না পারলে আমার কি কি ক্ষতি হয়ে যায়

To prove that a language is not regular one will follow the below steps. (By Contradiction)

1. Assume that A is regular.
2. It will have a pumping length p.
3. All strings longer than p can be pumped $|s| \geq p$.
4. Now find a string 's' in A such that $|s| \geq p$.
5. Divide s into three parts x, y, z
6. Show that $xy^iz \notin A$ for since i
7. Show that none of these can satisfy all the 3 pumping conditions at the same time.
8. s can not be pumped == contradiction.

Examples:

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

Prove that A is not a regular language.

Pumping length = p

so $s = a^p b^p$ [lets take $p = 7$]

$= aaaaaaabbbbbbb$

Case: 1 y is in the a part

aa, aaaa, abbbbbbb,
x y z

Case: 2 y is in the b part

aaaaaaabb, bbbb, b
x y z

Case-3. y is the combined position of a and b

aaaaa, aabbbbbbb,
x y z

যদি তোমার ভাষা [1] $xy^iz \in A$ for every $i \geq 0$ [2] $|y| > 0$ [3] $|xy| \leq p$ হলে Regular language.

এখন condition check করবো।

Case 1:

Cond 1 $xy^iz \Rightarrow xy^2z$ [let $i=2$]

$= aa\ aaaaaaaa\ abbbbbbb$

$a = 11$ $b = 7$

$a \neq b$

So condition false

আমাদের এ Language টি হচ্ছে যেখানে a এর সংখ্যা b এর সংখ্যা সমান কিন্তু এখানে $a \neq b$ তাই এটি A Language belong করে না।

Condition 2

$|y| > 0$

$4 > 0$

second condition true

Condition 3 $|xy| \leq p$ or $6 \leq 7$

Third condition true.

Case: 2

Condition 1

$xy^i z \Rightarrow xy^2 z$

$= aaaaaaabb\ bbbbbbbb\ b$

$a = 9 \quad b = 11$

$a \neq b$

First condition false.

Condition 2

$|y| > 0$

$q > 0$

Second condition true.

Condition 3

$|xy| \leq 7$

$13 \leq 7$ [Not Possible]

False.

Case 3

Condition

$xyz = x^a y^b z$

$= \underbrace{aaaaa}\ aabbaabb\ \underbrace{bbbbb}$

$a = 9 \quad b = 9$

A and B are equal but look at the pattern it does not fit in A so condition false.

2nd condition $|y| > 0$

$4 > 0$

True

3rd condition $|xy| \leq p$

$9 \leq 7$ [Not Possible]

False.

So A is not a regular language.

Example 2

Using pumping lemma prove that the language $A = \{yy \mid y \in (0,1)^*\}$ is not a regular language.

$A = \{yy \mid y \in (0,1)^*\}$

The set of all string that can be formed using the symbol 0 and 1

It means that The first part of the string should be same as the second part of the string

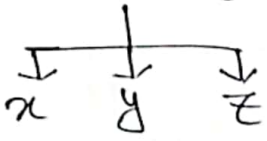
$\frac{01}{y}\ \frac{01}{y}$ First half second half 1st এবং 2য় 2য়

Proof

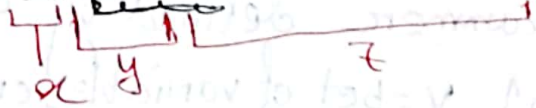
Assume that A is regular.

Pumping length $= p$.

$s = 0^p 1 0^p 1$ [lets take $p=7$]



Case 1: 0000000 1 0000000 1



Case 2:

Condition 1 $xy^1z = x^0y^2z$

$= 00\ 00000000\ 0100000001$

The first half of the string is not equal to the second half of the string.

Condition 2

$|y| > 0$

$4 > 0$

True

Cond 3

$|xy| \leq \phi p$

$10 \leq 7$

Condition is false.

Context free Grammers and Context free language

In formal language a theory, a context free Language is a Language generated by some context free grammers. The set of all CFL is identical to the set of languages accepted by Pushdown automata.

↓

Context free Language

CFL > Regular language

Context free grammers defined by 4 tuples

$G = \{V, \sum_{T}, S, P\}$ V = set of variables or non terminal symbol

Σ = set of terminal symbol

S = Start symbol P = Production Rule.

Context free grammare has a production rule of the form

$A \rightarrow \alpha$

where $\alpha = \{V \cup \Sigma\}^*$ and $A \in V$

α also could be empty symbol or anything from terminal and not terminal sybol.

Ex For generating a language that generates equal number of a's and b's in the form of $a^n b^n$, the context free grammare will be defined as

$G_1 = \{(S, A), (a, b), (S \rightarrow aAb, A \rightarrow aAb \mid \epsilon)\}$

- (S, A) → set of non terminal symbol
- (a, b) → set of terminal symbol
- S → aAb → start symbol
- A → aAb | ε

$S \Rightarrow aAb$

$\rightarrow aaAbb$ (by $A \rightarrow aAb$)

$\rightarrow aaaAbbb$ (n n n)

$\rightarrow aaabbb$ (by $A \rightarrow \epsilon$)

$\rightarrow a^3 b^3$

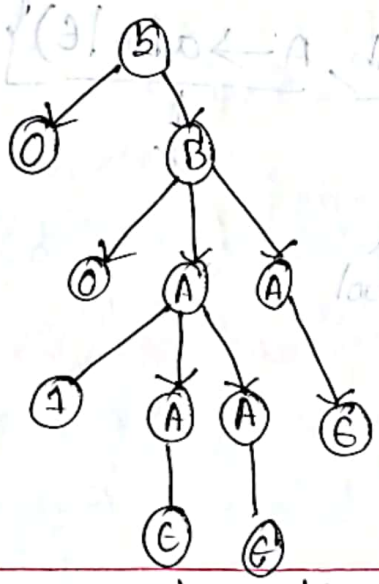
$\rightarrow a^n b^n$

Derivation Tree/ Parse Tree:

A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a context free grammar.

For the grammar $G_1 = \{V, T, P, S\}$ where $S \to 0B$

$A \to 1AA/\epsilon$, $B \to 0AA$



Root vertex: Must be labelled by the start symbol

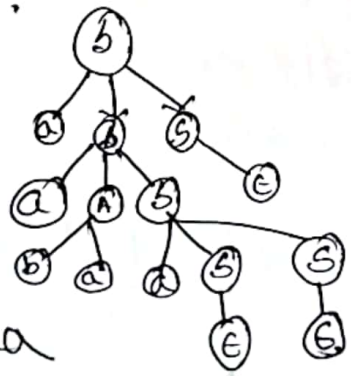
vertex: Labelled by the non terminal symbol.

Leaves: Labelled by the Terminal symbols or ϵ.

Generating string aabaa from the grammar $S \to aAS/a/S\epsilon$ $A \to Sba$

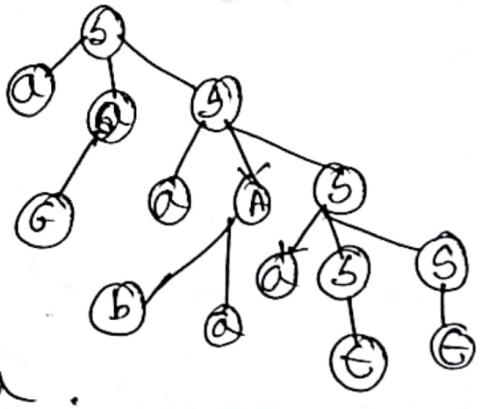
Two types derivation tree left or right.

LDT is obtained by applying production to the leftmost variable in each step.



aabaa

RDT is obtained by applying production to the rightmost variable in each step.



aabaa.

AB Ambiguous grammar → A grammar is said to be Ambiguous if there exists two or more derivation trees for a string w (that means two or more of left derivation trees)

Ex: $G_1 = (\{S\}, \{a+b, +, *\}, P, S)$ where P consists of

$S \to S+S \mid S*S \mid a \mid b$ The string $a+a*b$

$S \to S+S$
$\to a+S$
$\to a+b*S$
$\to a+a*S$
$\to a+a*b$.

$S \to S*S$
$\to S+S*S$
$\to a+S*S$
$\to a+a*S$
$\to a+a*b$.

Remove unit production from CFG1

① $A \to AB$, $A \to a$, $B \to C \mid b$, $C \to D$, $D \to E$, $E \to a$

$B \to C$, $C \to D$, $D \to E$ } unit production

② Remove unreachable variables

$A \to AB$
$A \to a$
$B \to a \mid b$

bo/n: $A \to AB$, $A \to a$, $B \to a \mid b$, $C \to a$, $D \to a$, $E \to a$

② $S \to aA|B$
$A \to ba|bb$
$B \to A|bba$

$S \to B$
$B \to A$
$S \to B \to A$

So/n

$S \to aA|ba|bb|bba$
$A \to ba|bb$
$B \to ba|bb|bba$

Remove unreachable variables.

$S \to aA|ba|bb|bba$
$A \to ba|bb$.

(Unit production বাদ দেওয়া হয়েছে।)

Null Production Removed / Remove null production.

① $S \to ABC$
$A \to aA|\epsilon$
$B \to bB|\epsilon$
$C \to C$

Directly or indirectly কে কে Null value produce করছে।

$A \to \epsilon$
$B \to \epsilon$ } Directly করছে

~~$S \Rightarrow ABC$~~

এরা দুইটা। Remove করতে হবে

Removing $A \to \epsilon$
$S \to ABC|BC$
$A \to aA|a$
$B \to bB|\epsilon$
$C \to C$

Removing $B \to \epsilon$
$S \to ABC|BC|AC|C$
$A \to aA|a$
$B \to bB|b$ $C \to C$.

②

$S \to ABAC$ $\quad A \to \epsilon$

$A \to aA \mid \epsilon$ $\quad B \to \epsilon$

$B \to bB \mid \epsilon$

$C \to C$.

Removing $A \to \epsilon$

$S \to ABAC \mid BAC \mid ABC \mid BC$

$A \to aA \mid a$

$B \to bB \mid \epsilon$

$C \to C$

Removing $B \to \epsilon$

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AC \mid AC \mid C$

$A \to aA \mid a$

$B \to bB \mid bB$

$C \to C$

③ $S \to aS \mid A$ $\quad A \to \epsilon$

$A \to \epsilon$ $\quad S \to \epsilon$ — Indirectly.

Removing $A \to \epsilon$

$S \to aS \mid \epsilon$

Removing $S \to \epsilon$

$S \to aS \mid \epsilon \mid a$

CFG to

Chomsky Normal Form:

① $S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

~~Right variable~~

1st step:

Starting variable এর R.H.S এ থাকলে নতুন

Starting variable নিতে হবে।

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

2nd step.

Remove null production

$B \rightarrow \epsilon$, $A \rightarrow \epsilon$

Removing $B \rightarrow \epsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a$

$A \rightarrow B \mid S \mid \epsilon$

$B \rightarrow b$

Removing $A \rightarrow \epsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \rightarrow B \mid S \mid \epsilon$

$B \rightarrow b$

Step 3

Remove unit production

$S_0 \rightarrow S$

$S \rightarrow S$

$A \rightarrow B$

$A \rightarrow S$

} unit production

এগুলো Remove করতে হবে।

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

Step 4 — দুই এর অধিক থাকলে দুইটা বানাইতে হবে।

CNF

$A \rightarrow a$

$A \rightarrow BC$

শুধুমাত্র দুইটা variable এর concatanation থাকবে।

$X \rightarrow SA$ মনে মনে ধরবো।

$S_0 \rightarrow AX \mid aB \mid a \mid SA \mid AS$

$S \rightarrow AX \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AX \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

$X \rightarrow SA$

(aB) এ form টা CNF এ নাই তাই step 5 এ যেতে হবে।

$Y \rightarrow a$ — মনে মনে ধরে নিবো।

Step 5

$S_0 \rightarrow AX \mid YB \mid a \mid SA \mid AS$

$S \rightarrow AX \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AX \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

$X \rightarrow SA$

$Y \rightarrow a$.

Pumping for context free ~~grammar~~ Language

prove a language is not context free.

Suppose A is a context free language so, A has a pumping length P such that any string s where $|s| \geq p$ may be divided into 5 pieces $s = uvxyz$ such that the following condition are true.

1. uv^ixy^iz is in A for every $i \geq 0$.
2. $|vy| > 0$
3. $|vxy| \leq p$

এই 3 টা condition true হইলে CFL

Steps to follow.

(i) Assume A is a context free.

(ii) It has to have

Prove that $L = \{a^N b^N c^N \mid N \geq 0\}$ is not a context free language.

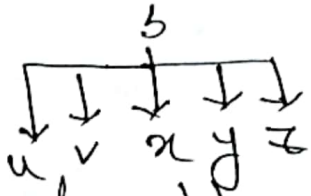
Sol: Here $N = 3$ $L = a^3 b^3 c^3 = aaa\ bbb\ ccc$

Final we assume that, L is a CFL, so it must have a pumping length P.

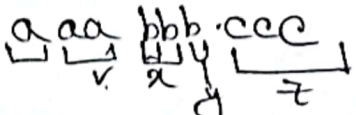
Let $P = 3$

So $s = a^3 b^3 c^3 = aaa\ bbb\ ccc$

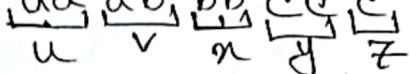
Now divided s into five parts.



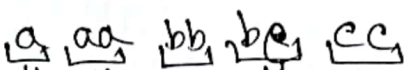
Case 1: v and y each contain only one type of symbol.



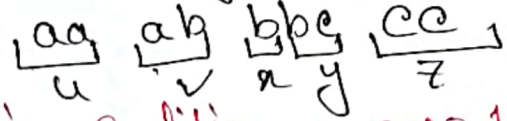
Case 2: v can have multiple types of symbol.



Case 3: y can have multiple types of symbol



Case 4: Both v and y can have multiple types of symbol.



Check Pumping Condition on case 1.

Condition-1: $uv^i xy^i z \in A$

Suppose $i = 2$

Condition 2: $|vy| > 0$

$3 > 0$ true

Condition 3: $|vxy| \leq p$ aabbb so $|vxy| = 5$ pumping length is 3. false. So it is not a cfl

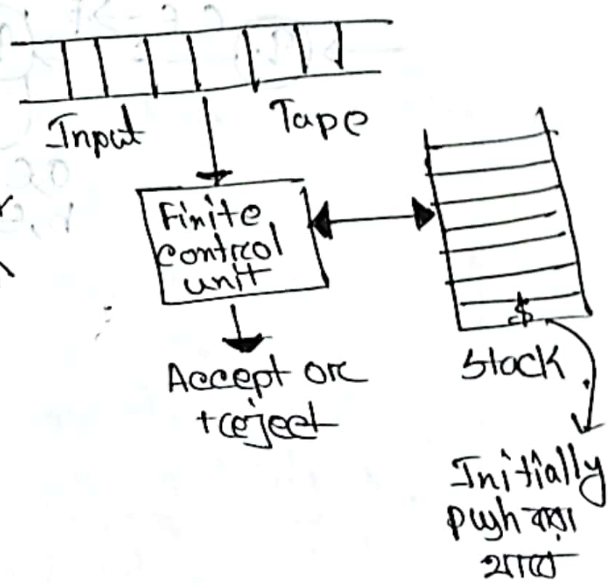
FSM + Stack — Alphabet trace করে।

Push Down Automata (PDA) - recognise CFL

Formal definition: 6 tuples = tuple($Q, \Sigma, \Gamma, \delta, q_0, F$)

- Q — Finite set of states
- Σ — Input Alphabet
- Γ — Stack Alphabet (যা যা push করবো তা Stack Alphabet)
- δ — $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow (Q \times \Gamma_\epsilon)$
- $q_0 \in Q$ is the start Alphabet
- $F \subseteq Q$ is the set of accept Alphabet

$L = \{0^n 1^n \mid n \geq 0\}$.



push ↓

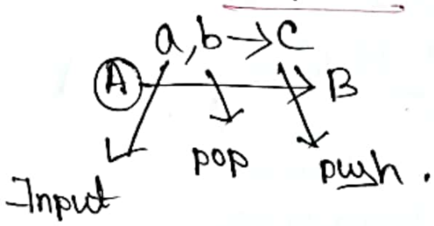
pop ↑

Transition Function

$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow (Q \times \Gamma_\epsilon)$

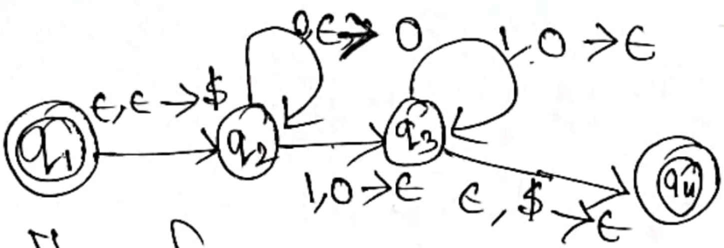
$(q, a, x) \rightarrow (q_1, 8)$

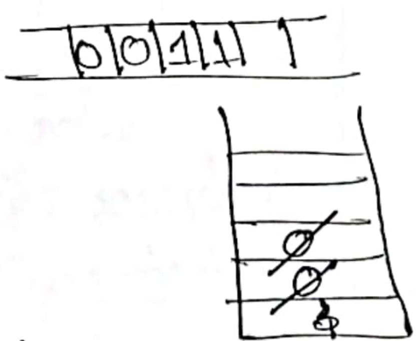
- q — State
- a — Input Symbol
- x — Stack Alphabet poped
- q_1 — New State
- 8 — Pushed Alphabet



Initial State অর্থাৎ Final STATE হবে।

Construct PDA that accepts $L = \{0^n 1^n \mid n \geq 0\}$





Transaction function

$(q_1, \epsilon, \epsilon) \rightarrow (q_2, \$)$

$(q_2, 0, \epsilon) \rightarrow (q_2, 0)$

$(q_2, 1, 0) \rightarrow (q_3, \epsilon)$

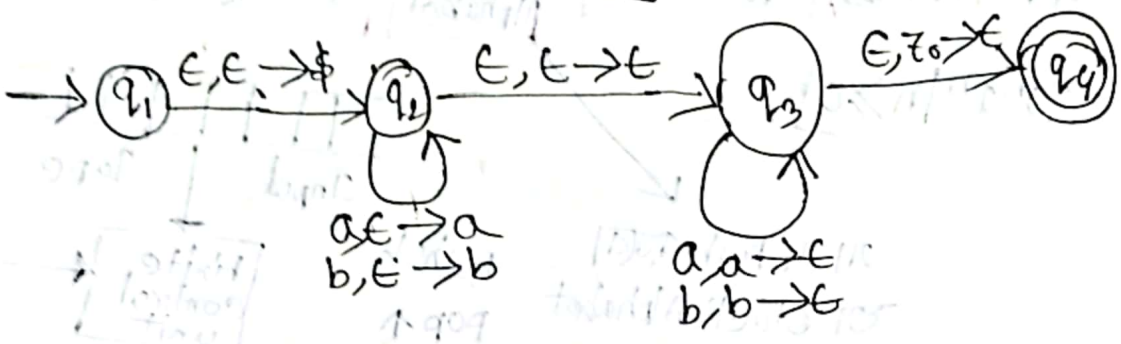
$(q_3, 1, 0) \rightarrow (q_3, \epsilon)$

$(q_3, \epsilon, \$) \rightarrow (q_4, \epsilon)$

Shirajul Islam C213040

Let's Construct a PDA that accepts Even palindromes of the form.

$$L = \{ WW^R \mid W = (a+b)^+ \}$$



Turing Machine: - Accepts recursively enumerable language.

- A tape

↓ Tap Head. — Cursor: যেখানে আছে তা Refer করে

Blank symbol (Input symbol এ belong করবে না)

Tape Alphabet $\Sigma = \{0, 1, a, b, x, \sqcup\}$

Operation

(i) Read/scan symbol below the tape head.
(ii) Update/write " " " " "
(iii) Move the tape head one step Left.
(iv) " " " " " " Right.

Rules of Operation 1

~~→ Read the current symbol. symbol.~~
→ Read the corresponding symbol.
→ update the same cell.
→ move exactly ~~one~~ one cell right or left.

Read করতে না চাইলে সেটা নিয়ে

Rules of operation 2

It's similar to a finite state machine.

Initial state.
Finite States (two final states) → The accept state
→ The reject state.

Computation can either.

* HALT and Accept
* HALT and Reject
* Loop (the machine fails to HALT)

Formal Definition of Turing Machine.

A turing machine can be ~~either~~ defined as ^ set of 7 tuples. $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$

$Q \rightarrow$ Non empty set of states.

$\Sigma \rightarrow$ Non empty set of symbols.

$\Gamma \rightarrow$ Non empty set of tape symbols.

$\delta \rightarrow$ Transition function defined as

$$Q \times \Sigma \rightarrow \Gamma \times (R \times L) \times Q$$

i.e $\delta(q_0, a) \rightarrow (q_1, Y, R)$

q_0 → Initial state

b → blank symbol

F → Set of final states (Accept or reject)

Design turing machine for $L = 0^N 1^N$

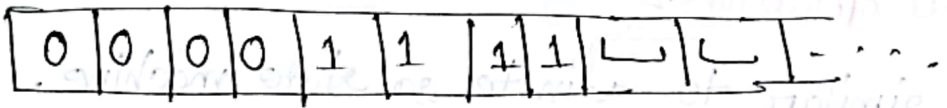
~~N=3~~

~~000111~~

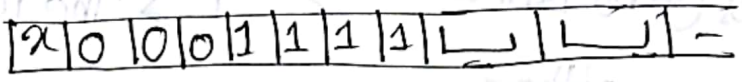
Sol^n: Suppose $N = 4$, so $L = 0^4 1^4$ so the string will be

0000 1111

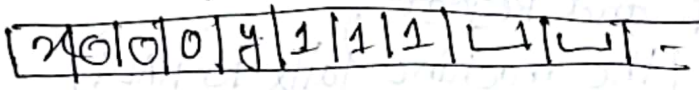
Now the tape will be like



Step 1 Change the first 0 to x



Step 2 move right to find the next 1 change it to y



Step 3 Move left to find the leftmost 0 and change it to x

x	x	0	0	y	1	1	1	␣	␣	. . .

Step 4: Again move right to find the first 1 and change it to y.

x	x	0	0	y	y	1	1	␣	␣	. . . .

Repeat the process until the tape become free from 0 and 1

x	x	x	x	y	y	y	y	␣	␣	. . .

If you failed to discuss this position that means the language is not accepted.