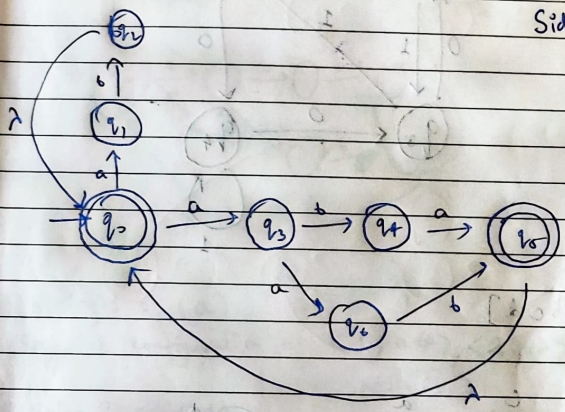
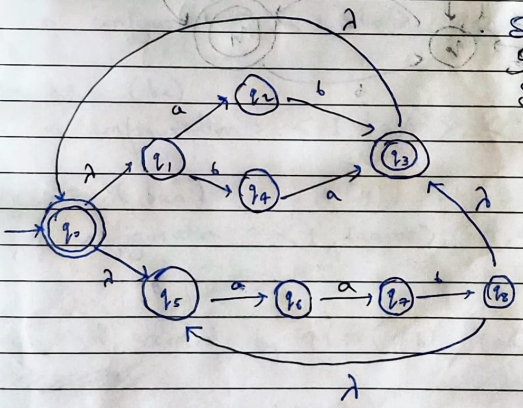
FORMAL LANGUAGE AND AUTOMATA THEORY CS302

Siddharth Sanskrityayan
1901CS75

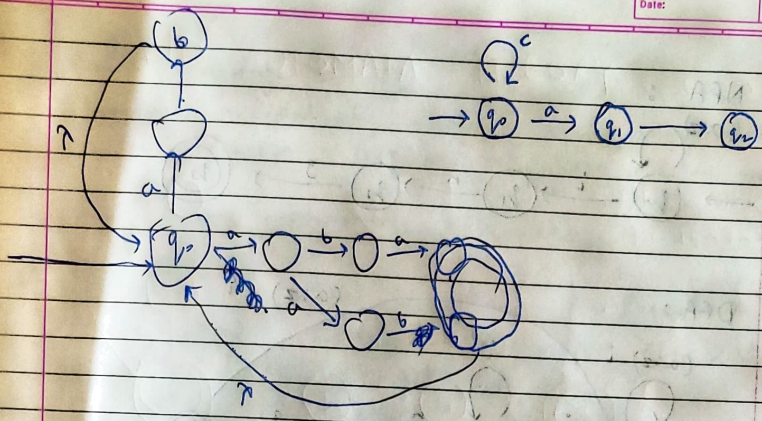
1a)



b)



x+y=z
x+y
x+y=z
x+y=z



4) i) string 1 = (a)
Seq. of configuration = <1, a#>, <2, #>

string 2 = (ab)
Seq. of configuration = <1, ab#>, <2, b#>, <2, #>

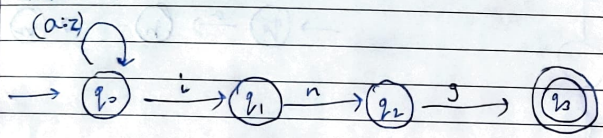
string 3 = (ba)
Seq. of configuration = <1, ba#>, <4, a#>, <3, #>

string 4 = (baa)
Seq. of configuration = <1, baa #>, <4, aa#>, <3, a#>, <2, #>

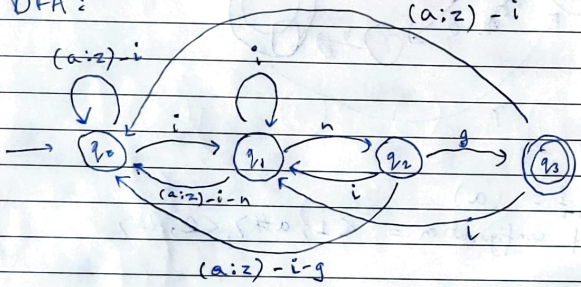
ii) Strings not accepted by M: (b), (aa), (bb), (aab)

iii) Q = {1, 2, 3, 4}
Σ = {a, b}
q_0 = {1}
F = {2, 3}

5) NFA :



DFA :



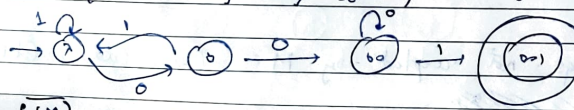
DFA : $M(Q, \Sigma, \delta, q_0, F)$

states, input alphabet, transition fn, initial state, accepting state

$\delta(q_1, a) = q_2$

Language accepted: $L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$

eg: $L(M) = \{$ all binary strings containing substring 001 $\}$



$\overline{L(M)}$ = change accepting states to non-accepting & vice versa

NFA eg: $L(M) = \{10\}^*$

Lang accepted by DFA = Lang accepted by NFA = regular language

Regular Expression :

Regular Language = Language accepted by DFA
= Language accepted by NFA = Language generated by regular expression = Language generated by regular grammar

For regular expressions r_1 and r_2

also regular

$L(r_1 + r_2) = L(r_1) \cup L(r_2)$
$L(r_1 . r_2) = L(r_1) L(r_2)$
$L(r_1^*) = (L(r_1))^*$
$L((r_1)) = L(r_1)$

Precedence
$* > . > \cup$

eg: $L((a+b.c)^*) = (L(a+bc))^*$
$= (L(a) \cup (L(b) . L(c)))^*$
$= (\{a\} \cup \{b\}.\{c\})^*$
$= \{a, bc\}^*$
$= \{\lambda, a, bc, aa, abc, bca, \ldots\}$

$L(\phi) = \phi$
$L(\lambda) = \{\lambda\}$
$L(a) = \{a\}$

eg: all strings containing substring 00

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

$L(0+1)^* = \{0,1\}^*$ = any string formed using any no. of 0 & any no. of 1

Formal Grammar

Page No.:
Date:

* Phase-structure : eg: <sentence> → <noun phrase> <predicate>
<noun phrase> → <article> <noun>
<predicate> → <verb>

<noun> → dog

Variable or non-terminal symbols of vocabulary — production rule — terminal symbols of vocabulary

PSG : $G = (V, T, S, P)$ — actual symbols of (eg dog) language

vocabulary, terminals, start, productions (rules)

$T \subseteq V$

Non terminals $N \equiv V - T$ — atleast one non-terminal on left side

* Language →

let $G(V, T, S, P)$

$L(G)$ → language generated by G is the set of all strings of terminals that are derivable from starting state S.

$$L(G) = \{ w \in T^* \mid S \Rightarrow^* w \}$$

Example

$G = (V, T, S, P)$

$T = \{a, b\}$, $V = \{a, b, S\}$

$P =$ $S \to aSb$; $S \to \lambda$

$S \Rightarrow aSb \overset{S \to \lambda}{\Rightarrow} ab$ | $\lambda, ab, aabb, \dots$

$\Downarrow S \to aSb$

$aaSbb \Rightarrow aabb$

$\Downarrow$

$aaaSbbb \Rightarrow aaabbb$

$\vdots$

$L = \{ a^n b^n : n \geq 0 \}$

Gift Sidd Sidd Gift

* PSG types:

Type 0 → unrestricted

Type 1 → Context sensitive PSG:

if $b \to a$ then $|b| \leq |a|$ $\vee$ $a = \lambda$

Type 2 → Context free PSG:

if $b \to a$ then $|b| = 1$ $(b \in N)$

Type 3 → Regular PSG:

if $b \to a$ then $a \in T$ $\vee$ $a \in TN$

* Linear Grammar: at most one variable at the right side of production

$S \to Ab$
$A \to aAb$
$A \to \lambda$

$S \to aSb$
$S \to \lambda$

$S \to SS$ → two variables
$S \to aSb$
$S \to \lambda$
X

↳ Right linear: $A \to xB$ where x is a string of terminals
$A \to x$

↳ Left linear: $A \to Bx$
$A \to x$

* Regular grammar ≡ left linear or right linear
* Regular grammar generate regular Language

Example:

$S \to Aab$
$A \to Aab \mid B$
$B \to a$

$L(G) = \{aab, aaab, aaaab, \ldots\} = a(ab)^*$
$L(G) = \{aab(ab)^*\}$

* Pumping lemma:

If A is a regular language, then $\exists$ a integer $p > 0$ (pumping length) : for any string $s \in A$ with $|s| \geq p$,

$s = xyz$ where

i) for each $i \geq 0$, $xy^iz \in A$
ii) $|y| > 0$
iii) $|xy| \leq p$

Page No.:
Date:

eg $L = \{ w w^R : w \in \Sigma^* \}$ $\Sigma = \{a, b\}$

Assume L is regular,

$\therefore L$ is infinite we can apply pumping lemma

Let m be the critical length

Let w be a string : $w \in L$ & length $|w| \geq m$

$$w = a^m b^m b^m a^m = xyz$$

with lengths $|xy| \leq m$, $|y| \geq 1$

$w = xyz =$ (a a ... a | a b ... b | b ... b a ... a)

xy | z

$\therefore y = a^k$, $1 \leq k \leq m$

From pumping lemma, $xy^i z \in L$ $\forall i = 0, 1, 2, \ldots$

(must hold for all $i \geq 0$)

Then $xy^2z \in L$

$\hookrightarrow a^m b^m b^m a^m . a^k$

$= a^{m+k} b^m b^m a^m \in L$ $k \geq 1$

$\hookrightarrow \notin L$

contradiction $\rightarrow$ L is non regular

Context free grammar :

All productions are of form $A \rightarrow S$

variable, variable & terminals

eg: $L = \{a^n b^n : n \geq 0\}$

is a context free language since

context free grammar $S \rightarrow aSb \mid \lambda$

generates $L(G) = L$

eg $S \rightarrow aSb \mid SS \mid \lambda$ $\rightarrow$ matched parenthesis

(( ) (( )) ( )) $a^*(\quad b^*)$

Context - free Languages

M T W T F S S
Page No.:
Date:
YOUVA

Grammar : $G = (V, T, S, P)$
variables, terminal symbols, start variable, productions

Productions are of the form $A \to S$
(variable) (variable & terminals)

eg: $S \to aSb \mid \lambda$

Context free
Regular

$V = \{S\}$ $T = \{a, b\}$ $S = \{S\}$

$P = \{ S \to aSb,\ S \to \lambda \}$

$L(G) = \{ w : S \overset{*}{\Rightarrow} w,\ w \in T^* \}$
* → sequence of steps
w → string of terminals or λ

For the eg $L(G) = \{ a^n b^n ; n \geq 0 \}$

Derivation tree (Parse tree)

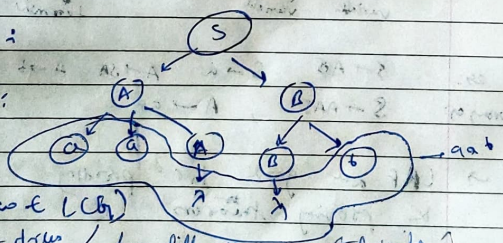
eg $S \to AB$, $A \to aaA \mid A$ $B \to Bb \mid \lambda$

Let us consider string aab

Leftmost derivation: $S \to AB \to aaAB \to aaB \to aaBb \to aab$

Rightmost derivation: $S \to AB \to ABb \to Ab \to aaAb \to aab$

give same derivation tree :



Ambiguous grammar:

Context free grammar G is ambiguous iff a string $w \in L(G)$ has 2 diff. derivation trees / two different leftmost derivations

eg: $E \to E+E \mid E*E \mid (E) \mid a$

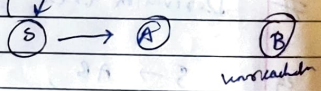
Simplifying context free grammar:

→ remove ε transitions (eg $A \to \varepsilon$)
→ remove unit production (eg : $A \to B$)
→ remove useless productions (unreachable or which never terminate)

(if any one of the variable in a production is useless then the production is also useless.

eg: $S \to aS \mid A \mid C$ useless variable
$A \to a$
$B \to aa$ ✗ not reachable from S
$C \to aCb$ ✗ not terminate

$S \to aS \mid A$
$A \to a$



Chomsky Normal form:

Prodⁿ rule form:

$A \to BC$ or $A \to a$
(variable, variable) (terminal)

eg: $S \to AS$ $S \to a$ $A \to SA$ $A \to b$

wrong eg: $S \to AAS$, $A \to aa$

→ CNF are good for parsing & proving theorem.

eg) $S \to aA$ $(B \to b)$
$A \to a$ $B \to bb$
$A \to B$
Substitute a $A \to a$ prod
$S \to aA \mid aB$
$A \to a$
$B \to A \mid bb$ $B \to bb$
Substitute $B \to A$
$S \to aA \mid aB \mid aa$
$A \to a$.
$B \to bb$

M T W T F S S
Page No.:
Date:
YOUVA

Context free Grammar → CNF conversion

→ remove nullable variables (by removing 2 transitions)
→ remove unit prod^n
→ remove useless variable (optional)
→ for every symbol 'a' (terminal), new var. T_a with prod^n $T_a \to a$

* $A \to C_1 C_2 \cdots C_n$ ⇐ $A \to C_1 V_1$,
$V_1 \to C_2 V_2$
;
$V_{n-2} \to C_{n-1} C_n$

* $A \to a$ unchanged

eg: $S \to ABa$, $A \to aab$, $B \to Ac$

$S \to A V_1$, $V_1 \to B T_a$, $A \to T_a V_2$, $V_2 \to T_a T_b$, $B \to A T_c$
$T_a \to a$, $T_b \to b$, $T_c \to c$

Greibach Normal form (GNF):

prod^n have form

$A \to a V_1 V_2 \cdots V_k$, $k \geq 0$
(a: terminal; $V_1 V_2 \cdots V_k$: variables)

ex: $S \to CAB$, $A \to aA \mid bB \mid b$, $B \to b$
wrong: $S \to abSb$, $S \to aa$

* GNF easy; good for parsing strings
but difficult to find GNF

Conversion from context free to CNF:

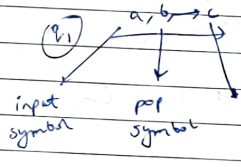
$S \rightarrow ab\,Sb$
$S \rightarrow aa$

$\Rightarrow$

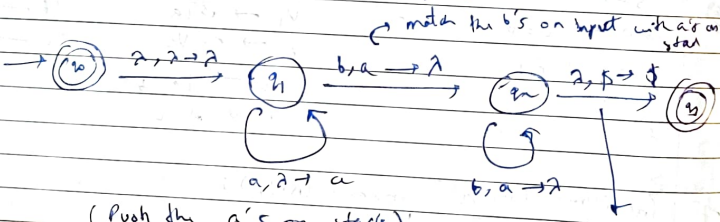
$S \rightarrow a T_1 S T_1$
$S \rightarrow a T_a$
$T_a \rightarrow a \qquad T_b \rightarrow b$

Pushdown Automata PDAs :

Input string → states ↔ stack



eg: $L(M) = \{a^n b^n ; n \geq 0\}$



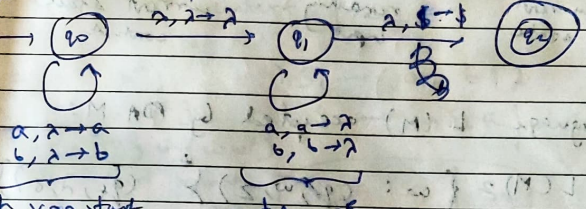
* A string is accepted if the input is consumed & the last state is an accepting state.
(stack content doesn't matter)

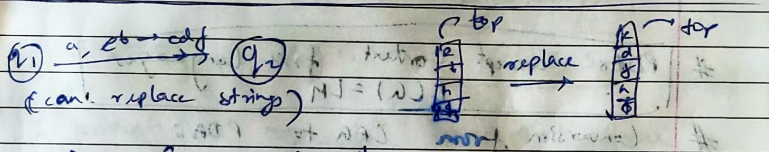
Computation

$(q_0, aabb, \$) \vdash (q_1, aabb, \$) \vdash (q_1, abb, a\$)$
$(q_0, bb, aa\$) \vdash (q_2, b, a\$) \vdash (q_2, \lambda, \$) \vdash (q_3, \lambda, \$)$

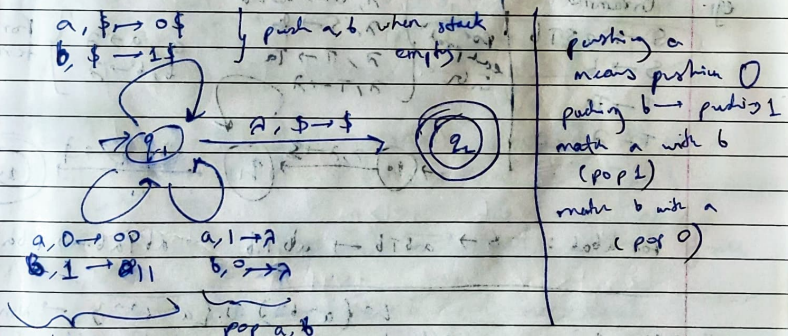
eg ii) $L(M) = \{ v v^R : v \in \{a,b\}^* \}$

guess middle of input



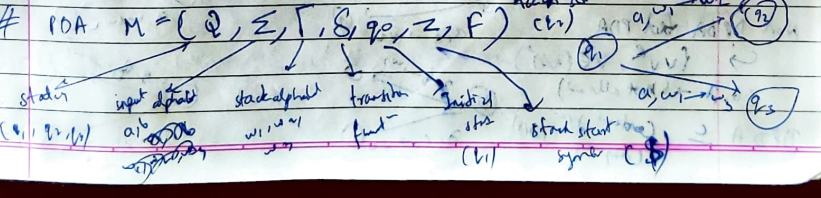


eg iii) $L(M) = \{ w \in \{a,b\}^+ : n_a(w) = n_b(w) \}$



$\delta(q_1, a, \omega_1) = \{(q_2, \omega_2), (q_3, \omega_3)\}$

PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$



* Instantaneous description : (q, v, s)

eg $(q_1, bbb, aaa\$)$

- q_1 : current state
- bbb : remaining input
- $aaa\$$: current stack contents

* Language $L(M)$ accepted by PDA M:

$$L(M) = \{ w : (q_0, w, z) \overset{*}{\vdash} (q_f, \lambda, s) \}$$

q_0 : initial state, q_f : final state

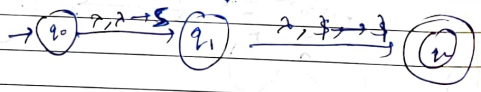
PDAs accept context free languages
$L(G) = L_M$

Conversion from CFG to PDA:

eg: Grammar
$S \to aSTb$
$S \to b$
$T \to Ta$
$T \to \lambda$

For each production in G:
- $\lambda, S \to aSTb$
- $\lambda, S \to b$
- $\lambda, T \to Ta$
- $\lambda, T \to \lambda$

For each terminal in G:
- $a, a \to \lambda$
- $b, b \to \lambda$



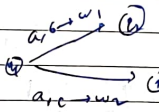
for abab : $S \to aSTb \to abTb \to abTab \to abab$

DPDA :

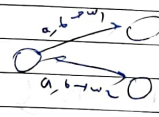
eg: $L_M = \{a^n b^n : n \geq 0\}$

$L = \{a^n b^n\} \cup \{a^n c^n\} \to$ Non DPDA

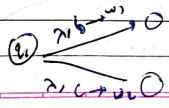
Allowed

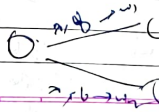


Not Allowed



Non DPDA but PDA
↳ $\{vv^R, v \in \{a,b\}^*\}$
$(b, \lambda \to b$ Not allowed)





DPDA $\subset$ Context free Language (PDA)

M T W T F S S
Page No.:
Date:
YOUVA

Properties of Context free languages :

i) Let L_1 → context free L_2 → context free
$L_1 \cup L_2$ → context free

eg: $L_1 = \{a^n b^n\}$ $S_1 \rightarrow aS_1b \mid \lambda$
$L_2 = \{ww^R\}$ $S_2 \rightarrow aS_2a \mid bS_2b \mid \lambda$
$L = \{a^n b^n\} \cup \{ww^R\}$
↳ New start symbol: $S = S_1 \mid S_2$

ii) L_1, L_2 is context free (concatenation).
$L = \{a^n b^n\}\{ww^R\}$: $S \rightarrow S_1 S_2$

iii) L is context free $\Rightarrow$ L^* is context free

eg: $L = \{a^n b^n\}$ $S_1 \rightarrow aS_1b \mid \lambda$
$L' = \{a^n b^n\}^*$ $S \rightarrow S_1 S \mid \lambda$

iv) $L_1 \cap L_2$ is not necessarily context free
eg: $L_1 = \{a^n b^n c^m\}$ (context free) $L_2 = \{a^n b^m c^n\}$
$L = L_1 \cap L_2 = \{a^n b^n c^n\}$ ⟹ not context free

v) L is context free ⟶ $\bar{L}$ is not necessarily context free

$\bar{L_1} \cup \bar{L_2} = L_1 \cap L_2 = \{a^n b^n c^n\}$ → not context free

vi) L_1 → context free L_2 → regular
$L_1 \cap L_2$ → context free

eg: i) $L = \{a^n b^n : n \neq 100 \quad , n \geq 0\}$
is context free?

$L_1 = \{a^n b^n ; n \geq 0\}$ → context free
$L_2 = \{a^{100} b^{100}\}$ → regular
$\overline{L_2} = \{(a+b)^*\} - \{a^{100} b^{100}\}$ → regular

$\therefore L_1 \cap \overline{L_2} = L$ = context free

ii) $L = \{w : n_a = n_b = n_c\}$ is not context free?

$L \cap \{a^* b^* c^*\} = \{a^n b^n c^n\}$

$\{a^* b^* c^*\}$ → regular
$\{a^n b^n c^n\}$ → not context free

answer: context free
contradiction

$\Rightarrow L$ is not context free

Pumping lemma for Context free language:

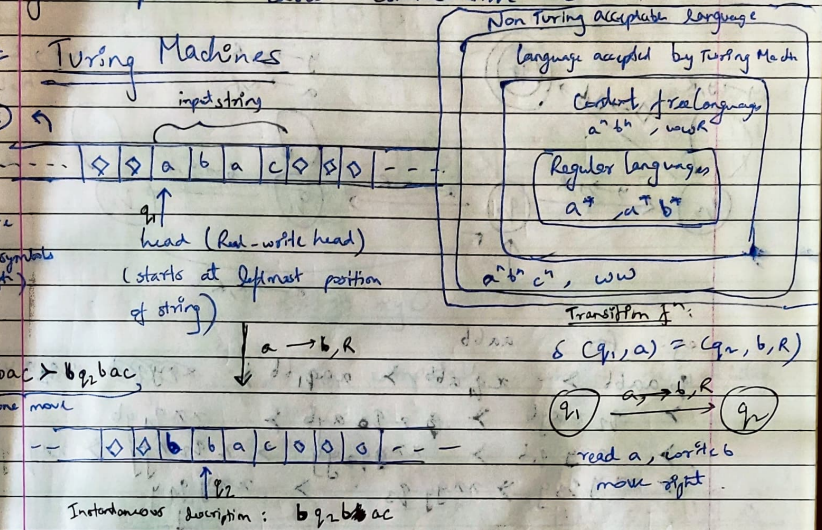
For infinite context free language L, $\exists$ an integer m (critical length) such that for any string $w \in L$, $|w| \geq m$
we can write $w = uvxyz$ with
$|vxy| \leq m$ and $|vy| \geq 1$ then it must be
$uv^i xy^i z \in L$ for all $i \geq 0$

eg: Non-context free: $\{a^n b^n c^n : n \geq 0\}$
$\{vv : v \in \{a,b\}^*\}$
$L = \{a^{n!} : n \geq 0\}$, $\{a^{n^2} b^n : n \geq 0\}$

Context free: $\{a^n b^n ; n \geq 0\}$
$\{ww^R : w \in \{a,b\}^*\}$

* For proving a language to be non-context free, assume context free, use pumping lemma & show for all possible cases contradiction occurs.

Turing Machines



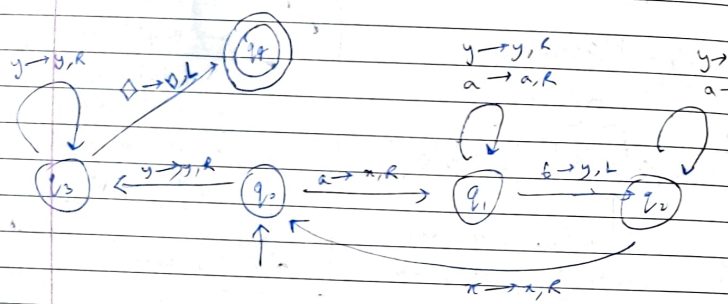
* machine halts in a state if there is no transition to follow

* accepting states has no outgoing transitions. The machine halts and accepts. (not necessary to scan all the symbols of string)

* Accept the string when machine halts in an accept state.
Reject input string when either:
i) machine halts in non-accepting state or
ii) machine enters infinite loop

* Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, \diamond, f)$
(states, input alphabet, tape alphabet, transition fⁿ, initial state, blank, accepting states)

eg: $a^n b^n : n \geq 1$ $\quad \Sigma = \{a, b\}$



eg. for aabb

$q_0 aabb \vdash x q_1 abb \vdash x a q_1 bb \vdash x q_2 a y b \vdash q_2 x a y b \vdash x q_0 a y b \vdash x x q_1 y b \vdash x x y q_1 b \vdash x x q_2 y y \vdash x q_2 x y y \vdash x x q_1 y y \vdash x x y q_1 y \vdash x x y q_3 y \vdash x x y y q_3 \vdash x x y q_4 y$

Halt and accept

* Language L is Turing Recognizable or Turing Acceptable or Recursively Enumerable if

$L(M) = \{ w : q_0 w \overset{*}{\vdash} x_1 q_f x_2 \}$

where q_0 is initial state & q_f is an accepting state

Computing functions with Turing Machines:

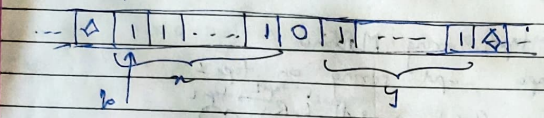
* A funct^n f is computable if there is a turing machine M :

$q_0 w \overset{*}{\vdash} q_f f(w)$ $\quad \forall w \in D$ (domain)

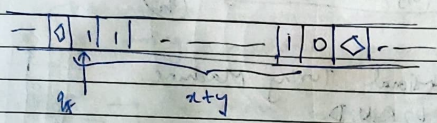
$q_0 w$: initial config

$q_f f(w)$: final config

* eg: $f(x,y) = x+y$ is computable where x, y are integers

Start: 

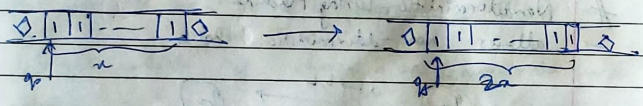
Binary representation
5 = 11111
2 = 11

Finish 

$q_0 x 0 y \vdash^* q_f$ ~~[illegible]~~ $xy0$

* eg2: $f(x) = 2x$ is computable where x is an integer

$q_0 x \vdash^* q_f xx$

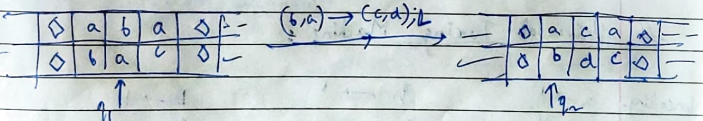


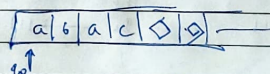
Variations of Turing Machine:

i) Stay Option : left, Right, stay : possible head moves

ii) Multiple Track Tape:

One tape



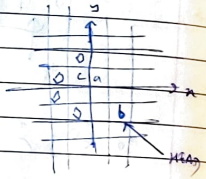
iii) Semi-Infinite tape: 

When head moves left from the border, it returns to same position.

iv) Off-line Machine
(Input string appears on input file only)

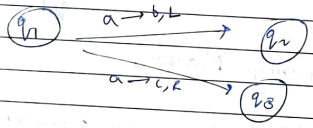
v) Multi-tape Turing Machine:
Two tapes, input string on tape 1

vi) Multidimensional Turing Machine:
Moves : L, R, U, D
Head position : +2, -1



* All these machines have the same power but may not have same speed

Nondeterministic Turing Machine:
→ allows non-deterministic choices



* Nondeterministic machines have the same power with Standard Turing Machine

Universal Turing Machine:

Encoding
a → 1
b → 11
c → 111
:

$q_1 → 1$
$q_2 → 11$
$q_3 → 111$
:

L → 1
R → 11

UTM : (3 tapes)
Tape 1 : description of M
Tape 2 : Tape contents of M
Tape 3 : State of M

M T W T F S S
Page No.:
Date:
YOUVA

eg: $\delta(q_1, a) = (q_2, b, L)$ $\quad \delta(q_2, b) = (q_3, c, R)$

Encoding: 10101101101 00 1101101110111101
(Tape 1) ↑ separator

* Each turing machine is a binary string of 0's & 1's
The set of turing machines forms a language

Countable set → one to one correspondence with +ve integers.
eg: odd/even integers, rational nos, $\{a,b,c\}^+$, $\{a,b\}^*$

* Enumerator:
* An enumerator for S (set of strings / language) is a Turing Machine that generates (prints on tape) all the strings of S one by one & each string is generated in finite time

* A set S is countable if there is an enumerator for S.

* Thm: The set of all Turing Machines is countable
(generate the next binary string of 0's & 1's in proper order)

* If S is an infinite countable set, then powerset 2^S of S is uncountable

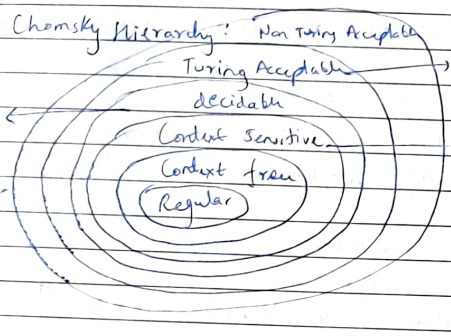
* There exist languages which ~~are~~ can't be ~~recognized~~ accepted by any turing machine
(Turing machines are countable, languages are uncountable)

* let $S = \{a,b\}^*$, $X = \{L_1, L_2, L_3, \ldots\}$ → language accepted by turing machine (countable)

$X \subset 2^S$ ($X \neq 2^S$)
2^S → set of all languages (uncountable)

* $X \subset 2^{S} \Rightarrow \exists L' \in 2^{S}$ and $L' \notin X$
$\Rightarrow L'$ can not be described by any algorithm / turing m/c

Chomsky Hierarchy:



→ generated by unrestricted grammar ($V \rightarrow \gamma$)

→ accepted by linear-bounded automation ($V \rightarrow \gamma$ & $|V| \leq |\gamma|$)

don't (there is a language which is context sensitive but not decidable)

Decidable language:

→ A language L is decidable if there is a turing machine (decider) M which accepts L and halts on every input string. (also called recursive language)

→ $w \in L \Rightarrow M$ halts in accept state
$w \notin L \Rightarrow M$ halts in non-accept state (doesn't loop forever)

→ Every decidable language is Turing acceptable

* Thm: If L is decidable, then its complement $\bar{L}$ is also decidable

* Undecidable language → no decider i.e each turing machine that accepts L does not halt on some input string.

* Thm: A language is turing acceptable iff there is an enumerator for it.

M T W T F S S
Page No.
Date
YOUVA

* Language L is decidable then there is an enumerator for it.

Proof: Let $\tilde{M}$ be the enumerator that generates all strings in proper order

Repeat: i) $\tilde{M}$ generates a string w
ii) M checks if $w \in L$ (M is the decider for L)
YES: print w to output
NO: ignore w

a
b
aa
ab
ba
bb
aaa
.
.

* A computational problem is decidable if the corresponding language is decidable
We say that the problem is solvable.

eg: Do DFAs M_1, M_2 accept the same language?
↳ problem

EQUAL$_{DFA}$ → Corresponding language $\{ \langle M_1, M_2 \rangle : M_1 \& M_2$ are DFAs that accept same language$\}$

Decider for the language:
Let L_1 → language of DFA M_1
L_2 → " " " M_2
Construct DFA $L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$
If $L(M) = \phi \Rightarrow L_1 = L_2$
else $L_1 \neq L_2$
($L(M) = \phi$ is determined whether there is path from initial state to any accepting state)

* For undecidable language, the corresponding problem is undecidable (unsolvable)

→ Membership problem : Does Turing Machine M accept w?
or $w \in L(M)$?
This is undecidable (answer may be given for some input instances)

→ Halting problem is undecidable:
Does Turing Machine M halt while processing input string w?

→ Both membership problem & halting problem is Turing acceptable.

Reductions:

→ Problem X is reduced to Problem Y
⇒ If we can solve Y then we can solve X

→ Language A is reduced to Language B
$w \in A \iff f(w) \in B$
f → computable function (deterministic turing machine which for any string w computes f(w))

→ Thm: If i) language A is reduced to B
ii) Language B is decidable
Then A is decidable.
(eg: $EQUAL_{DFA}$ is reduced to $EMPTY_{DFA}$)