

03/28

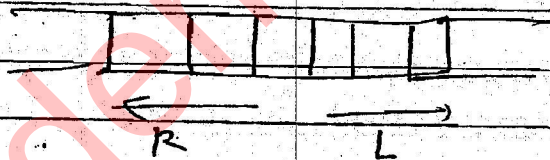
Date : / /

- May be FA or infinite Automata.

CSL - Context Sensitive language
• Requires ~~PDA~~ Turing Machine

 ~~$0^n 1^n 2^n$~~ 
$$TM = (FA + R/W) + R \rightarrow L$$

- In TM, you can move in both direction right as well as left.



✓ ✓ ✓
000 111 222

Move for first match

- come back find for 1 untick value

0 0 1 1 2 2

and so on

if no symbol left then its is accepted.

Download all Notes and papers from StudentSuvidha.com

- Turing Machine has much power but it is not fully used by CSL. then we can use LBA (Lower bound Automata)
- LBA is a special TM where length of tape is bounded with length Left end Marker & Right end Marker



i.e., here working space of the tape is limited i.e. limited M/M.

- You can also move right or left acc to required.
- Here M/M of ^(LBA) automata is depending upon input. if input expands, M/M also expands. Due to which it is able to solve infinite Automata as well. So, it is diff. from FA because FA will have M/M which is independent of input.
- In FA, M/M size depends upon states in FA.
- LBA will not be able to solve the problem which are recursive because seq. of M/M is not clear through input.

For Recursive problem LREC we use Halting Turing Machine (FA + R/W + R/L)

- the limitation with HTM is it ~~can~~ must halt, it must never get stuck in infinite loop.
- ~~It never halt as~~

A Machine halt if it get stuck in Right-Left Move, for

never ~~halt~~ hang of machine, ^{what} we did, for each move we enter into new state.

RE - a language in which for some input it may halt or may not halt.

After REC, system comes into state where it ~~may~~ ^{may} halt or may not halt.

Ex- Software

• It uses RE (FA + R/W + R/E)

↳ it may or may not halt

• Turing Machine has power the equivalent to logic. no machine powerful than TM exist.

church turing thesis, "Turing Machine has same power to logic"

• if Logic exists for a problem, TM will work/exist.

Machine has two uses -

1. Compiler (to recognize special language)
2. Transducer (to perform computation)

Acceptors

The problems which a turing machine cannot handle is

Not RE

↓ the problem which can not be represented by logic.

the whole classification is 2^{Σ^*} as it covers all the aspect of given string set

↓ power set of Σ^*

possible on Σ .

FA is ^{not} RE & RE is not FA

13 Representation of Languages

1. Listing Method

2. Statement Method

3. Set-Builder Method

Informal Method of Representation

(suitable for Human but not for system)

for not RE

4. Machine

5. Grammar

6. Regular expression

suitable for CSL, CFL, RE (all, every)

Formal

Least power as only can represent RL

Ex - even no. of zero

3. Set-Builder Method - $L = \{0^{2n} \mid n \geq 0\}$

2. Stmt - $\{ \text{even no. of zero} \}$

1. Listing = $\{ \epsilon, 00, 0000, 000000, \dots \}$

4. Machine



① - final state

→ start/initial state

(practice)

5. Grammar -

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

$S \rightarrow 00S \mid \epsilon$

$V = \{S\}$

$T = \{0\}$

$P: S \rightarrow 00S \mid \epsilon$

start symbol

6. Regular Expression a.e. = $(00)^*$

14.

Operation on Languages- $U, \cap, -, L^c, L_1 \oplus L_2$

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

$$L_1 = \{10, 0, 11\}$$

$$L_2 = \{11, 01\}$$

$$L_1 \cup L_2 = \{11, 0, 11, 01\}$$

$$L_1 \cap L_2 = \{11\}$$

$$L_1 - L_2 = \{10, 0\}$$

$$L_1^c = \bar{L}_1 = \Sigma^* - L_1$$

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$$

$$\bar{L}_1 = \{\epsilon, 1, 00, 01, 000, \dots\}$$

$$L_2 = \Sigma^* - L_2$$

$$= \{\epsilon, 1, 0, 00, 10, 000, \dots\}$$

Ques

$$\Sigma = \{0\}$$

$$L = \{0^{2n} \mid n \geq 0\}$$

$$\bar{L} = \Sigma^* - L$$

$$= \Sigma^* - L = \{0^{(2n+1)} \mid n \geq 0\}$$

$$L \cup \bar{L} = \Sigma^*$$

$$L \cap \bar{L} = \emptyset$$

The complement of a finite language is always infinite = True
 The complement of an infinite language is always finite = False
 (may or may not be finite)

If the $\bar{L}$ is infinite then L may or may not be finite.

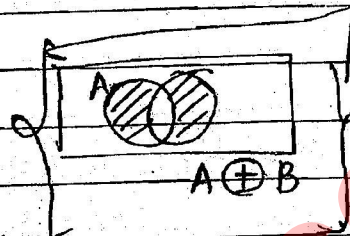
If the $\bar{L}$ is finite then L must be infinite.

$$L_1 - L_2 = L_1 - (L_1 \cap L_2)$$

$$= \cancel{L_1} - \cancel{L_1} \cap (L_1 \cap L_2)$$

$$L_2 - L_1 = \{10\}$$

$$A \oplus B = (A - B) \cup (B - A)$$



$$L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1)$$

$$= \{10, 0, 01\}$$

belong to L_1 only & L_2 only but not both

$$L_1 \oplus L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)$$

$$L_1 \oplus L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)$$

15. Concatenation & Reversal

Reversal of a language $L^R = \{w^R \mid w \in L\}$

$$L_1 = \{01, 100, 00\}$$

$$L^R = \{10, 001, 00\}$$

$$1. |L^R| = |L|$$

$$2. \left[w_1 = w_2 \text{ iff } \{ w_1^R = w_2^R \} \right]$$

To Reverse an infinite language,

$$L = (01)^* 11$$

$$L^R = 11(10)^*$$

$$L = (01)^* 10$$

$$L^R = 01(10)^*$$

$$L = (10+00)^* 100$$

$$L^R = 001(01+00)^* \checkmark$$

$$L^R = 001(00+01)^* \checkmark$$

$$L^R \neq 001(10+00)^* \times$$

as + is commutative

EXCELLENT

+ - commutative

Page No.

Date: / /

$$(a+b)^* = \{a, ab, abb, baq\}$$

$$(ab+ba)^* \neq abba \text{ extra term}$$

ues - $L^R = L$ iff L is palindrome
Maybe true or may not be.

as $L = \{10, 01\}$

$$L^R = \{01, 10\} \Rightarrow \text{true}$$

but if

$$L = \{101, 010\}$$

$$L^R = \{101, 010\} \Rightarrow \text{true}$$

true for if

but iff is used for definition so 1st condition get violated as 10 and 01 are not palindrome but $L^R = L$

Concatenation - $L_1 \cdot L_2 = \{uv \mid u \in L_1 \& v \in L_2\}$

$$L_1 = \{01, 10, 11\}$$

$$L_2 = \{1, 0\}$$

it must cover every combination

$$L_1 \cdot L_2 = \{011, 101, 111, 010, 100, 110\}$$

ie.

must not contain duplicate

$$|L_1 \cdot L_2| \neq |L_1| \times |L_2|$$

$$|L_1 \cdot L_2| \leq |L_1| |L_2|$$

because there may be some duplicate combination

$$L_1 = \{01, 10, 11\}$$

$$L_2 = \{1\}$$

EXCELLENT

$$L_2 L_1 = \{ uv \mid u \in L_2, v \in L_1 \}$$

$$L_2 L_1 = \{ 101, 110, 111, 001, 010, 011 \}$$

Also $|L_2 L_1| \neq |L_1 L_2|$

1. $L_2 L_1 \neq L_1 L_2$ (Not commutative)

2. $L_1(L_2 L_3) = (L_1 L_2) L_3$ (but associative)

Ex - $L_1 = \{ 0^n \mid n \geq 0 \}$
 $L_2 = \{ 1^n \mid n \geq 0 \}$

$$L_1 \cdot L_2 = \{ \begin{array}{l} (a) \{ 0^n 1^n \mid n \geq 0 \} \\ (b) \{ 0^m 1^n \mid m, n \geq 0 \} \\ (c) \{ 0^m 1^m \mid m \geq 0 \} \end{array}$$

as $L_1 = \{ 0000 \}$
 $L_2 = \{ 11 \}$ as both are local variable

$$L_1 \cdot L_2 = \{ 0^m 1^n \mid m, n \geq 0 \}$$

$$L_1 = \{ 0^n \mid n \geq 0 \}$$

$$L_2 = \{ 1^n 2^n \mid n \geq 0 \}$$

$$[L_1 \cdot L_2 = \{ 0^m 1^n 2^n \mid m, n \geq 0 \}]$$

as they both must be equal

Power of a language (L^n)

$$L^0 = \{\epsilon\} \quad (\text{as contributing to identity element})$$

$$L^1 = L$$

$$L^2 = L \cdot L$$

$$L^3 = L \cdot L^2 = L^2 \cdot L = L \cdot L \cdot L$$

$$L \cdot \phi = \phi$$

attaching nothing to L will result to ϕ

$$L^0 = \{\epsilon\}$$

$$L^1 = \{10, 01\}$$

$$L^2 = \{10, 01\} \cdot \{10, 01\}$$

$$= \{1010, 1001, 0110, 0101\}$$

To given string belong to L^2

- break into two parts

10 10

- if successful break i.e. the atom elmt is in L

else does not belong

$$x \in L^n \text{ iff } x = \dots$$

$$\bigvee_{L^n \text{ parts}} \epsilon_L$$

$$* \text{ if } L = \{01, 10, \epsilon\}$$

then

$$L^2 = 01 \text{ as } \left\{ \begin{array}{l} 01 \in L^2 \\ \text{as } 01 \cdot \epsilon \end{array} \right\}$$

if ϵ is there in L to check $x \in L^n$ divide it into ~~no~~ less than n parts. ~~if it works,~~

Ques

$L = \{0^n \mid n \geq 0\}$
 (a) $L^2 = \{0^n \mid n \geq 0\}$ ✓ as both n are independent local vari
 (b) $L^2 = \{0^{2n} \mid n \geq 0\}$ false ~~as if 0^{2n}~~

16. L^* & L^+

closure of L , $L^* = L^0 \cup L^1 \cup L^2 \dots$

(+)ve closure on L , $L^+ = L^1 \cup L^2 \cup L^3 \cup \dots$

Ex $L = \{01, 10\}$

$L^0 = \epsilon$

$L^1 = \{01, 10\}$

$L^2 = \{0110, 0101, 1001, 1010\}$

$L^* = \{\epsilon, 01, 10, 0110, 0101, 1001, 1010, \dots\}$

• to check given string belong to L^*

• break into any no of pieces (acc. to L).

as 01 10 10 01 01 11

✓ Do not belong.

• if $L = \{01, 101, 110\}$

01 101 00

Not po

• in L , if ϵ is there, it does not make any difference.

Ques $(L^*) = (\bar{L})^*$ (Not equal)
 $\downarrow$
 $\emptyset$ (false)

Ques - a. belonging of string to L^2 does not mean belonging to L^3 as if there is ϵ in L , it also belongs to L^3

Ques - $L^+ = L^* - \epsilon$ (false) (because L^+ can contain ϵ)

* L^+ can contain ϵ as L contains ϵ but it will not contain L^0 .

Ques $L^+ = L^* - \epsilon$ iff $\epsilon \in L$

i.e, if a language is a null free language.

Ques $L^+ = L^*$ if L contains ϵ

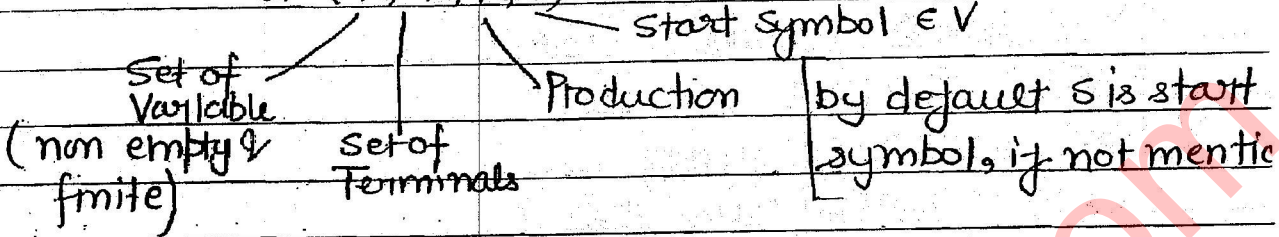
Grammar

if and iff under diff

Page No.
 Date: / /

Grammar is a 4-tuple

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



- In every grammar, atleast one variable is required.
- Variable - capitals
- Terminal - small - finite it can be empty.
- Empty - $S \rightarrow \epsilon$

• Production is also non empty.

$$|P| \geq 1$$

as for $\emptyset$, we need a dummy production.

$$S \rightarrow A$$

all 3 terminal variable & production must be finite.

- for a single grammar, many single machine is possible. but a minimal DFA, i.e no of states in a worst machine.

$$\text{Let } P = \{S \rightarrow \alpha \alpha S, S \rightarrow \epsilon\} \quad \text{if } P = \{S \rightarrow \alpha \alpha S, S \rightarrow \epsilon\} \quad |P| = 2$$

cardinality of P = No of production = 2

- for a single grammar, a single language is possible only.

Derivation

<

>

to represent variable
Backus Naur Representation

without Bracket, terminal

EXCELLENT

$\langle \text{Sentence} \rangle \rightarrow \langle \text{Noun phrase} \rangle \langle \text{verb} \rangle$
 $P := \{ \langle \text{sentence} \rangle \rightarrow \langle \text{Noun phrase} \rangle \langle \text{verb} \rangle$
 $\langle \text{Noun phrase} \rangle \rightarrow \langle \text{Adj} \rangle \langle \text{Noun} \rangle$
 $\langle \text{Adj} \rangle \rightarrow a / the$
 $\langle \text{Noun phrase} \rangle \rightarrow \text{boy} / \text{dog}$
 $\langle \text{verb} \rangle \rightarrow \text{walks} / \text{jump} / \text{run} \}$

$S \rightarrow \langle \text{sentence} \rangle$

$T = \{a, the, boy, dog, walk, jump, run\}$

$V = \{\langle S \rangle, \langle N \rangle, \langle A \rangle, \langle V \rangle\}$

Cardinality, $|P| = 9$

Derivation is useful to match the I/P string to the pattern i.e. for checking for compilation.

$\langle \text{sen} \rangle \rightarrow \langle \text{Noun} \rangle \langle \text{verb} \rangle$

$\rightarrow \langle \text{Adj} \rangle \langle \text{Noun} \rangle \langle \text{verb} \rangle$

$\rightarrow a \langle \text{Noun} \rangle \langle \text{verb} \rangle$

$\rightarrow a \text{ boy walks}$

When left variable getting substituted

- at one step of derivation, do substitute one variable only.
- A production can be used to many no. of times.
- A Grammar to generate infinite set, must contain recursion.

$S \rightarrow OS \rightarrow \text{Right Recursion.}$

$S \rightarrow SO \rightarrow \text{Left Recursion.}$

- ~~Atom~~ Ambiguity has no mean with left or Right recursion.
- Recursion will always not lead to infinite.

Ambiguity in C - Dangling else

Page No.

Date : / /

Theorem

A string w belongs to L iff
 $w \in L(G)$ iff $\exists$ derivation of w using P

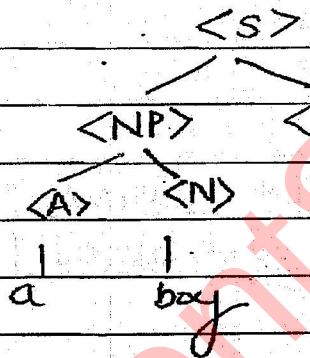
[iff & only if]

• Derivation of a string cannot contain any variable.

i.e. $w \in T^*$

• Derivation may or may not be unique for a given string. It can be at least 1.

• Ambiguity does not come due to multiple derivation but due to multiple derivation tree.



for derivation tree, no ambiguity due to diff. substitution becoz you do any side (Right or left) anytime.

• Ambiguity is related to semantic & derivation tree is related to syntax.

Derivation - Syntax
Derivation Tree - Semantics

EXCELLENT

$$S \rightarrow 00S | \epsilon$$

$$S \rightarrow 00S \rightarrow 0000S$$

$$\rightarrow 000000$$

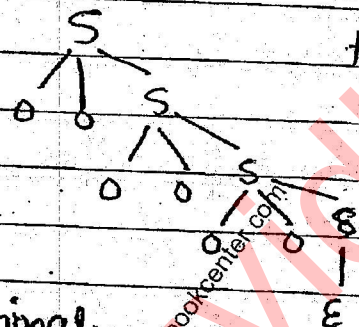
Sentential form - $(VUT)^*$

$$\begin{cases} 00S \\ 0000S \end{cases}$$

000000 - Sentence $[(T)^*]$

[A sentence is always a sentential form]

$$\text{yield} = 000000\epsilon$$

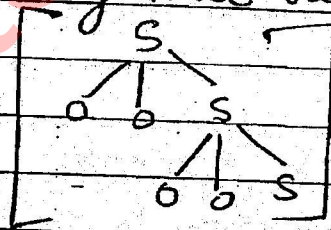


the yield of a given direct derivation tree is the sentence

- Every leaf node is terminal.

partial derivation tree - the yield can be sentential form

- It may have variable at leaf



Derivation Tree -

- Root must be start symbol
- Right derivation is correct
- yield must be sentence

Partial DT -

- Root may or may not be start symbol.
- yield may or may not be sentence.
- It may or may not be having variable at leaf.

$L(G_1) = L(G_2)$
↳ Undecidable problem

Page No.

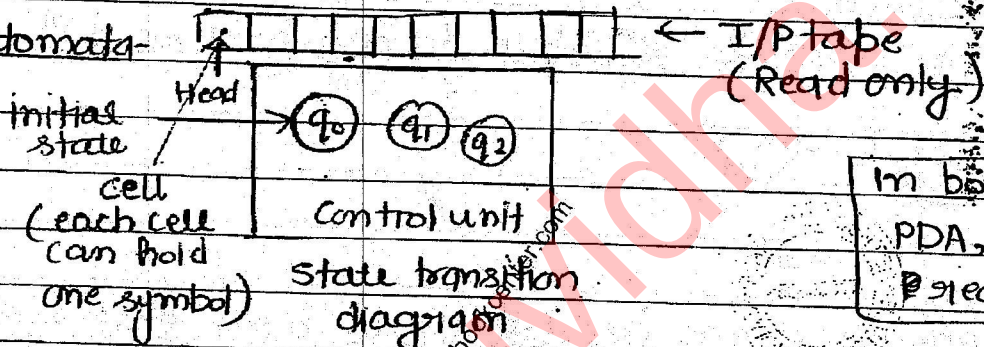
Date: / /

[A Derivation Tree is always a Partial DT but PDT is not always DT.]

Machine

- The equivalence of two grammar is decidable iff they are regular (both)
- checking ambiguity of a grammar is undecidable.

Finite Automata-



In both FA & PDA, tape is read only

- At end of tape, ϵ is placed.
- After Reading an I/P, it moves toward Right
- Read only head can move forward only

→ $\odot$ — final state (accepting state)

→ At end of tape, machine must halt.

if halt at non final state — string rejected
if halt at final state — string accepted.

• PDA is provided with stack of infinite size.

for transducer — FA and TM

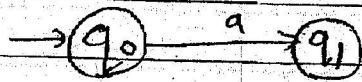
Mealy Machine Moore Machine

for acceptance — DFA & NFA

EXCELLENT

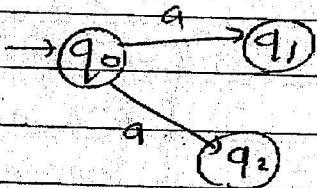
Deterministic Finite state Acceptance-

- single path i.e. for an I/P, only one O/P exist.
i.e. if you execute program multiple ~~at~~ time for one I/P, it must produce same result.



Non-Deterministic FA -

- for a single ~~off~~ Input, it may produce diff. O/P.



Machine work by divide itself ~~to~~ traces both path to move.

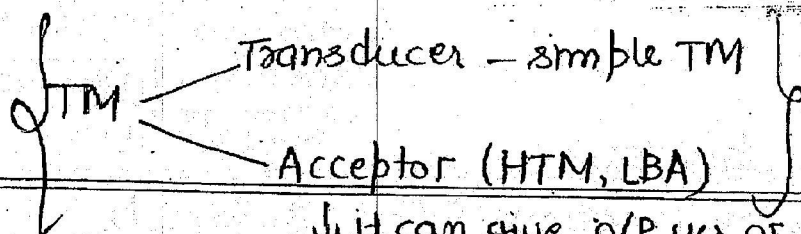
- It always produce a results if exist.

- It help to perform Backtracking to given problem by moving into many branch.
- It is easier to program.

- both DFA & NFA has same power of work.

DFA $\equiv$ NFA

- If a program is Require backtracking & searching, NFA is used to simulate that program.



Page No.

Date : / /

↓ it can give o/p yes or no whether string accepted or rejected.

Mealy

- FA can transduce simple f^n but TM can transduce every logically computable f^n .
- A logically computable f^n is also partially recursive function.

Mealy & Moore • both have same power

in Mealy machine - O/P depending on current state as well as input symbol.

in Moore Machine - O/P is only f^n of the current state it does not depend on input symbols.

• for a specific state, O/P is specified, no matter whatever the input symbol is.

• Mealy can be converted to moore & vice versa

• Mealy is more superior to moore as Mealy is easier to implement.

EXCELLENT