Toppersnotes

Unleash the topper in you

UGC-NET

Computer Science & Application

NATIONAL TESTING AGENCY (NTA)

PAPER – 2 || VOLUME – 3

THEORY OF COMPUTATION AND COMPILERS,
DATA COMMUNICATION AND COMPUTER NETWORKS,
ARTIFICIAL INTELLIGENCE (AI)

SIERRA
INNOVATIONS

Index

Unit 8: THEORY OF COMPUTATION AND COMPILERS

Unit 9: DATA COMMUNICATION AND COMPUTER NETWORKS

Unit 10: ARTIFICIAL INTELLIGENCE (AI)

THEORY OF COMPUTATION AND COMPILERS

Introduction

- Also known as automata Theory.
- its goal is to analyze the behaviour of machines & how they solve a problem.
- its most powerful model is Turing machine.

Formal language

- is a set of strings of symbols drawn from a finite alphabet.
- a formal language can be specified either by a set of rules that generates the language or by a formal machine that accepts the language
- it can be grouped into a series of successively large classes known as Chomsky hierarchy.

Non - Computational Problems

- it has no algorithm to solve its problems.
- eg Halting Problem

Russels's Paradox -

- also known as Russell's Antinomy

Regular Language Models

a language is regular if it can be expressed in terms of regular expression

- an expression is regular if;

1) ϕ is a regular expression for regular language ϕ

2) if a & b are regular expression, a+b is also regular expression with language {a, b}

3) if a & b are regular expression, a* b is also regular.

4) if a is regular, a* (0 or more times a) is also regular.

- ~~a language is~~

- a grammer is regular if it has rules of form $A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \varepsilon$

where ε is special symbol called NULL

Q1 which of the following languages over the alphabet {0, 1} is described by the regular expression?

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

(A) The set of all strings containing the substring 00

(B) The set of all strings containing at most two 0's

(C) The set of all strings containing at least two 0's.

(D) None of the above

Sol option C

∵ it says that it must contain atleast two 0. In regular expression, two 0 are present

Q2 which of the following languages is generated by given grammar?

$$S \rightarrow aS \mid bS \mid \epsilon$$

(A) $\{a^n b^m \mid n, m >= 0\}$

(B) $\{w \in \{a,b\}^* \mid$ w has equal no. of a's & b's$\}$

(C) $\{a^n \mid n>=0\} \cup \{b^n \mid n>=0\} \cup \{a^n b^n \mid n>=0\}$

(D) $\{a, b\}^*$

Sol option D

$\because$ it says that it can have any no. of a's & any no. of b's in any order.

Q3 The regular expression $0^*(10^*)^*$ denotes the same set as

(A) $(1^*0)^*1^*$

(B) $0+(0+10)^*$

(C) $(0+1)^*10(0+1)^*$

(D) None of these

Sol Two regular expressions are equivalent if languages generated by them are same

so $0^*(10^*)^* = (1^*0)^*1^*$

Deterministic Finite Automaton (DFA)

- In DFA, for each input symbol, one can determine the state to which the machine will move.
- it has finite no. ~~of~~ of states, the machine is called Deterministic Finite machine.

⇒ Formal Definition -

- DFA can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q = finite set of states
- Σ = finite set of symbols (alphabets)
- δ = transition function where $\delta: Q \times \Sigma \rightarrow Q$
- q_0 = initial state from where any i/p is processed ($q_0 \in Q$)
- F = is a set of final state of Q ($F \subseteq Q$)

⇒ Graphical Representation –

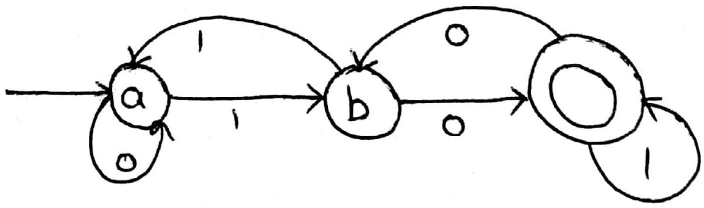
- DFA is represented by a digraph called state diagram.

1) vertices represent states
2) arcs labeled with an i/p alphabet shows the transitions.
3) Initial state is denoted by an empty single incoming arc.
4) Final state is indicated by double circle

eg
$Q = \{a, b, c\}$
$\Sigma = \{0, 1\}$
$q_0 = \{a\}$
$F = \{c\}$
δ is shown by this table ↓

Present State	Next State for i/p 0	Next State for i/p 1
a	a	b
b	c	a
c	b	c

sol



Non-Deterministic Finite Automaton (NDFA)

- For a particular input symbol, the machine can move to any combination of the states in the machine
- In NDFA, the exact state to which the machine moves cannot be determined.

⇒ Formal Definition:

1) Q = Finite set of states
2) Σ = finite set of symbols
3) δ = a transition function where

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

4) q_0 = Initial state where any i/p is processed $(q_0 \in Q)$
5) F = is a ~~finite~~ set of final state of Q $(F \subseteq Q)$

DFA	NDFA
(1) Transition from a state is to a single particular next state for each input symbol	(1) The transition from a state can be to multiple next states for each input symbol.
(2) Empty string transitions are not seen in DFA	(2) NDFA permits empty string transitions
(3) Backtracking is allowed in DFA	(3) Backtracking is not possible
(4) Requires more space	(4) Less space

Regular Languages

- it can be expressed with regular expression or deterministic/ non-deterministic finite automata or state machine.
- Regular languages are subset of set of all strings.
- RL are used in parsing & designing programming languages.

Regular languages & finite automata can model computational problems that require a very small amount of memory.

⇒ **Operations :**

A regular language can be represented by a string of symbols & operations,

1) Concatenation
2) Union
3) Kleene Star
4) Empty string
5) Language Notation

A language is said to be a regular language if some Finite state machine recognizes it

Union operation

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

Concatenation

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

Star

$$A^* = \{x_1 x_2 x_3 \ldots x_k \mid k \geq 0 \text{ and each } x \in A\}$$

Regular Expression Identities –

1) $\phi + R = R$

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

3) $\varepsilon R = R\varepsilon = R$

4) $\varepsilon^* = E$ and $\phi^* = E$

5) $R + R = R$

6) $RR^* = R^*R$

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

8) $\epsilon + RR^* = \varepsilon + R^*R = R^*$

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

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

11) $(P+Q)R = PR + QR$

Context Free Language

- CFG consisting of a finite set of grammar rules is a quadruple (N, T, P, S)
- N = set of non-terminal symbols
- T = set of terminals

$N \cap T = Null$

- P = set of rules

$P: N \rightarrow (N \cup T)^*$

ie LHS of the production rule P does have any right context or left context

- S = Start symbol

⇒ In CFG, all the production rules & symbols are not needed for the derivation of strings.

- Elimination of these productions & symbols is called simplification of CFGs.
- Simplification of CFG comprises of some steps:
  - ★ Reduction of CFG
  - ★ Removal of Unit Productions
  - ★ Removal of Null Productions

- CFG is a set of recursive rules used to generate patterns of strings.
- CFG describe all regular languages & more, but cannot describe all possible languages.
- a grammar can be used to describe the possible hierarchical structure of a program.
- CFG has four components:
  1) set of tockens, known as terminal symbols
  2) a set of nonterminals
  3) a set of productions
  4) a destination of one of the nonterminals as the start symbol.

Q ~~Let~~ The grammar can be defined as

$G = (V, \Sigma, P, S)$

In the given definition, what does S represents?

sol Starting Variable

explaination V = Finite set of variables

Σ = set of terminals

P = finite productions

S = Starting variable

Q which of the following statement is false?

a) Context free language is the ~~s~~ subset of context sensitive language

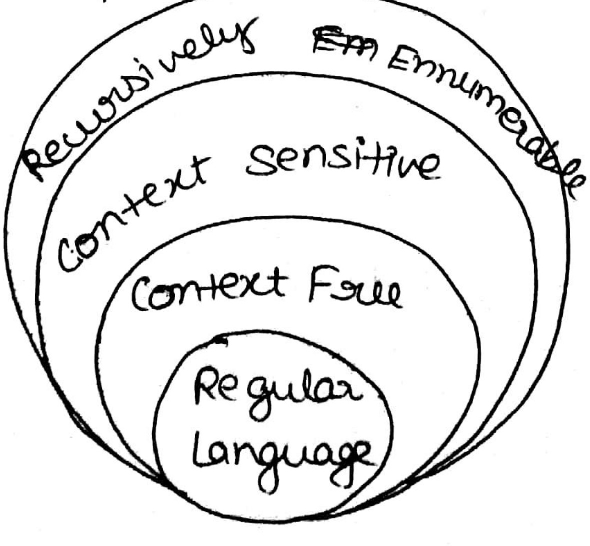
b) Regular language is the subset of context sensitive language

c) Recursively ennumerable language is the super set of regular language

d) Context sensitive language is a subset of CFL

sol (d)

Every regular language can be produced by CFG & context free language can be produced by context sensitive grammar & so on.



Q context - free grammar can be recognized by

(A) finite - State automation

(B) 2 - way linear bounded automata

(C) Push down automata

(D) Both (b) & (c)

sol (D) is correct option

Q The language L =(0n 1n 2n where n>0) is a

sol context - sensitive language

Q context - ~~sensi~~ free language are closed under -

sol union, kleene closure

Q A context free grammar G is in Chomsky normal form if every production is of the form

ans $A \rightarrow BC$ or $A \rightarrow a$

Chomsky Normal Form

- A CFG is in Chomsky Normal Form if the productions are in the following forms-

$A \rightarrow a$ — ①

$A \rightarrow BC$ — ②

$S \rightarrow \varepsilon$ — ③

where A, B, C are non-terminals & a is terminal.

Conditions -

- A non-terminal generating a terminal — ① explaination of ①
- A non-terminal generating two non-terminals — explaination of ②
- start symbol generating ε — explaination of ③