DATE | | PAGE

① Write a short note on Chomsky's Normal Form (CNF).

ⓐ CNF stands for Chomsky's Normal Form

ⓑ A CFG (context free grammar) is in CNF if all production rules satisfy one of the following conditions
- Start symbol generating ε, For eg: A → ε
- A non terminal generating two non-terminals. Eg S → AB
- A non terminal generating a terminal. Eg S → A

ⓒ For example:-

$G_1 = \{ S \to AB, S \to c, A \to a, B \to b \}$

$G_2 = \{ S \to aA, A \to a, B \to c \}$

Production rules of G_1 satisfy rules for CNF. So, $G_1 \to$ CNF.
However, G_2 production does not satisfy the rule S → aZ
contains terminal followed by non-terminal.
∴ G_2 is not a CNF.

ⓓ Steps to convert CFG to CNF -

Step 1:- Eliminate start symbol from RHS. If the start symbol T is at the RHS, create a new production as -

S1 → S, where S1 is new start symbol

Step 2:- In the grammar, remove the null, unit, useless productions

Step 3:- Eliminate terminals from RHS of the production if they exist with other non-terminals or terminals

Eg:- S → aA can be decomposed as: S → RA, R → a.

Step 4:- Elimate RHS with more than 2 non-terminals. For

Eg:- S → ACB can be decomposed as

S → RS

R → a

DATE | | PAGE

(2) Write a note on Greibach Normal Form (GNF).

(a) GNF stands for Greibach Normal Form

(b) A CFG (context free grammar) is in GNF if all the production rules satisfy one of the following conditions-

- A start symbol generating ε. For example, S → ε
- A non terminal generating a terminal. Eg: A → a
- A non terminal generating a terminal which is followed by any number of terminals. For eg: S → aASB

(c) For example -

G1 : { S → aAB | aB, A → aA | a, B → bB | b }

G2 : { S → aAB | aB, A → aA | ε, B → bB | ε }

The production rules of grammar G1 satisfy the rules specified for GNF, so the grammar G1 is GNF. However, the production rule of Grammar G2 does not satisfy the rules specified for GNF as A → ε and B → ε contains only ε (only Start symbol can generate ε).

∴ G2 is not in GNF

(d) Steps for converting CFG into GNF.

Step 1 :- Convert the given grammar into CNF if its not in CNF

Step 2 :- If the grammar exists left recursion, eliminate it.

Step 3 :- In the grammar, convert the given production rule into GNF form.

If any production rule in the grammar is not in GNF form, convert it.

DATE | | PAGE

(5) Write a brief about "sentential form" with reference to context free grammar

(a) A sentinal form is any string derivable from the start symbol. Thus, in the derivation of $a+a*a$, $E+T*F$ and $E+F*A$ and $F+a*a$ are all sentential forms as are E and $a+a*a$ themselves.

(b) Eg— Let $G=(V,T,P,s)$ be a CFG and $\alpha \in (V \cup T)^*$

If $S \xrightarrow{*} \alpha$

then, we say that α is a sentential form

(c) If $S \xRightarrow[lm]{} \alpha$, we say that α is a left sentential form, and if $S \xRightarrow[rm]{} \alpha$, we say that α is a right sentential form

(d) Note:- $L(G)$ is those sentential forms that are in T^*

(e) Consider the given CFG,

$(\{S, B\}, \{a, b\}, S, \{S \to aS, S \to B, B \to bB, B \to \lambda\})$

A derivation using this grammar might look like this:-

$S \Rightarrow aS \Rightarrow aB \Rightarrow abB \Rightarrow abbB \Rightarrow abb$

Each of $\{S, aS, aB, abB, abbB, abb\}$ is a sentential form.

Because this grammar is linear, each sentential form has at most one variable. Hence, there is never any choice about which variable to expand next.

(4) Justify how a Turing Machine can simulate a general purpose computer and vice versa.

1- (a) The Church-Turing thesis states that any sufficiently powerful computational model which captures the notion of algorithm is computationally equivalent to the Turing machine. This equivalence usually holds both a computability level and at a computational complexity level modulo polynomial reductions.

(b) In particular, it is shown that some models of computational perspective, general purpose analog computer is equivalent to computable analysis.

(c) Connection between this two models, form a computational complexity level, by showing that, modulo polynomial reductions computations of Turing Machines can be by GPACs, without the need of using more resource than those used in the original Turing computation, as long as we are talking about bounded computations.

(d) Simulating Turing Machine by computer -

(a) Simulation of the Turing Machine on a Digital Computer is a useful and practical tool not only for problem solving and validation of algorithms but also in the field of programming.

(b) TM are as powerful as the most sophisticated real-world programming languages, which is great, because formally proving anything about c, java would be difficult