**Chapter 8: CONVERSIONS – III**

**Topic – 1: Chomsky’s Normal Form (CNF)**

**Introduction**

* **Set of rules** from which **at least one** has to be satisfied by **each production rule** to be called as a **CNF**.

**CNF Rules**

* **Rule 1:** Start symbol generates **Ԑ**.
* **Rule 2:** A **terminal** generating **two non-terminals**.
* **Rule 3:** A **terminal** generating a **terminal**.

**Steps Involved**

* **Step 1:** If start symbol is at RHS of any terminal, create new start symbol.
* **Step 2:** Simplify the CFG.
* **Step 3:** Decompose productions where **terminal** & **non-terminal** exist together.
* **Step 4:** Decompose productions having **more than two** terminals.

**Example**

**Ques: Convert the given CFG grammar into CNF.**

**S 🡪 a | aA | B**

**A 🡪 aBB | Ԑ**

**B 🡪 Aa | b**

**Ans:**

**Here is no S at the RHS, so we will start with step 2 i.e. removing nulls.**

**After removal of Ԑ:**

**S0 🡪 S**

**S 🡪 a | aA | Aa | b**

**A 🡪 aBB**

**B 🡪 Aa | b | a**

**'a' is added to B as Ԑ of A is making aA in S as 'a' when null.**

**We replaced B with what it generated on S’s RHS. The same wasn’t done on A as it causing it to keep non-terminal values on both left & right side of the terminal.**

**Time to apply step 3 i.e. eliminating co-existing terminals & non-terminals:**

**S0 🡪 S**

**S 🡪 a | XA | AX | b**

**A 🡪 XBB**

**B 🡪 AX | b | X**

**X 🡪 a**

**Finally, we will simplify RHS of terminal A to have just two terminals:**

**S0 🡪 S**

**S 🡪 a | XA | AX | b**

**A 🡪 YB**

**B 🡪 AX | b | X**

**X 🡪 a**

**Y 🡪 XB**

**Topic – 2: Greibach Normal Form (GNF)**

**Introduction**

* Just like **CNF**, another kind of **normal form**.
* **At least one** of the three given rules must be satisfied for **each production**.

**GNF Rules**

* **Rule 1:** Only start symbol can generate **Ԑ**.
* **Rule 2:** A terminal generating just one **non-terminal**.
* **Rule 3:** A **terminal** generating just **one** **non-terminal**, which is followed by a number of terminals.

**Steps Involved**

* **Step 1:** Convert the grammar into **CNF** first.
* **Step 2:** Eliminate any **left recursion** if they exist.
* **Step 3:** Convert each production rule into **GNF** form.

**Example**

**Ques: Convert given grammar to GNF.**

**S 🡪 XB | AA**

**A 🡪 a | SA**

**B 🡪 b**

**X 🡪 a**

**Ans:**

**Given grammar is already in CNF so we will skip step 1.**

**No left recursion is there so we will also skip step 2.**

**Applying step 3, we will replace S:**

**S 🡪 XB | AA**

**A 🡪 a | XBA | AAA**

**B 🡪 b**

**X 🡪 a**

**It becomes:**

**S 🡪 aB | AA**

**A 🡪 a | aBA | AAA**

**B 🡪 b**

**X 🡪 a**

**We remove left recursion (A 🡪 AAA):**

**S 🡪 aB | AA**

**A 🡪 aC | aBAC**

**C 🡪 AAC | Ԑ**

**B 🡪 b**

**X 🡪 a**

**Now don’t ask how we got C.**

**Eliminating Ԑ:**

**S 🡪 aB | AA**

**A 🡪 aC | aBAC | a | aBA**

**C 🡪 AAC | AA**

**B 🡪 b**

**X 🡪 a**

**We put Ԑ in AAC to get AA.**

**Now we will replace terms with terminals coming first:**

**S 🡪 aB | aCA | aBACA | aA | aBAA**

**A 🡪 aC | aBAC | a | aBA**

**C 🡪 aCAC | aBACAC | aAC | aBAAC | aCA | aBACA | aA | aBAA**

**B 🡪 b**

**X 🡪 a**