UNIT II

Context free grammar

Context free grammar is a formal grammar which is used to generate all possible strings in a given formal language.

Context free grammar G can be defined by four tuples as:

1. G= (V, T, P, S)

Where,

G describes the grammar

T describes a finite set of terminal symbols.

V describes a finite set of non-terminal symbols

P describes a set of production rules

S is the start symbol.

In CFG, the start symbol is used to derive the string. You can derive the string by repeatedly replacing a non-terminal by the right hand side of the production, until all non-terminal have been replaced by terminal symbols.

Example:

L= {wcw^R | w € (a, b)*}

Production rules:

1. S → aSa
2. S → bSb
3. S → c Now check that abbcbba string can be derived from the given CFG.

1. S ⇒ aSa
2. S ⇒ abSba
3. S ⇒ abbSbba
4. S ⇒ abbcbba

By applying the production S → aSa, S → bSb recursively and finally applying the production S → c, we get the string abbcbba.

Capabilities of CFG

There are the various capabilities of CFG:

- Context free grammar is useful to describe most of the programming languages.
- If the grammar is properly designed then an efficientparser can be constructed automatically.
- Using the features of associatively & precedence information, suitable grammars for expressions can be constructed.
- Context free grammar is capable of describing nested structures like: balanced parentheses, matching begin-end, corresponding if-then-else's & so on.

- We have to decide the non-terminal which is to be replaced.
- We have to decide the production rule by which the non-terminal will be replaced.

We have two options to decide which non-terminal to be replaced with production rule.

Left-most Derivation

In the left most derivation, the input is scanned and replaced with the production rule from left to right. So in left most derivatives we read the input string from left to right.

Example: **Production rules:**

1. S = S + S
2. S = S - S
3. S = a | b |c

Input:

```
a - b + c
```

The left-most derivation is:

1. S = S + S
2. S = S - S + S
3. S = a - S + S
4. S = a - b + S
5. S = a - b + c

Right-most Derivation

In the right most derivation, the input is scanned and replaced with the production rule from right to left. So in right most derivatives we read the input string from right to left.

Example:

1. S = S + S
2. S = S - S
3. S = a | b |c

Input:

```
a - b + c
```

The right-most derivation is:

1. S = S - S
2. S = S - S + S
3. S = S - S + c
4. S = S - b + c
5. S = a - b + c