Context Free Grammar (CFG)

Derivation Tree or Parse Tree

Derivation process

There are two types of derivation process,

1. Left most derivation (LMD)
2. Right most derivation (RMD)

Left most derivation (LMD):

A derivation S ⇒ w is called a leftmost derivation if we apply a production only to the leftmost variable at every step.

Right most derivation (RMD):

A derivation S ⇒ w is called a rightmost derivation if we apply a production to the right most variable at every step.

Example: Consider a grammar G for the language, L = $\{a^{2n}b^{m}, m, n \geq 0\}$

G: S → AB, A → aaA | ε, B → bB | ε, find the LMD and RMD for string w = aab

Example: G: S □ S+S | S*S | a | b | c, find the LMD and RMD for string w = a + b * c

Ambiguous Grammar

A grammar G, is ambiguous grammar if ∃w ∈ L (G), such that w has > one Parse tree either using 2 LMD or 2 RMD i.e., a grammar G is ambiguous if there is more than one Parse Tree or LMD / RMD for a string w ∈ L (G).

Example: G: S S+S | S*S | a | b | c, Grammar G is ambiguous because there exist two different LMD for a string w = a + b * c

NOTE:

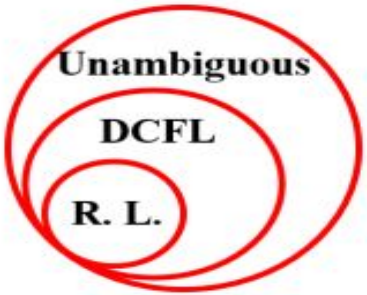
1- A grammar G is unambiguous if there exist exactly one parse tree or LMD / RMD for all the string, $w \in L(G)$

2- If a grammar G is ambiguous, it doesn't mean language (L) is ambiguous.

Example: State whether given grammar is ambiguous or not. G: E → E+E | E-E | id

NOTE:

- Regular Grammar:
  - Ambiguous (NFA)
  - Unambiguous (DFA)
- Regular language is always unambiguous.
- DCFL is always unambiguous language
- Ambiguity starts from CFL



Inherent ambiguous language:

Example: Check whether the given language $L = \{a^n \mid n \geq 0\}$ is an inherently ambiguous or not

Example: Given language L = $\{a^m b^n c^k \mid m, n, k \geq 1$, either m = n or n = k $\}$ is Inherent Ambiguous?

Note: If grammar is ambiguous then it is not suitable for any kind of parsing technique except backtracking and operator precedence parsing

Remove Ambiguity:

It is not always possible to convert an ambiguous grammar into an unambiguous grammar because ambiguity finding & removal both are undecidable

Remove Ambiguity by precedence & associativity rules:

An ambiguous grammar may be converted into an unambiguous grammar by implementing:

- Precedence Constraints
- Associativity Constraints

These constraints are implemented using the following rules:

Rule 1: The precedence constraint is implemented using the following rules:

- The level at which the production is present defines the priority of the operator contained in it.
- The top level of the production, the lower the priority of the operator.
- The bottom level of production, the higher the priority of the operator.

Rule 2: The associativity constraint is implemented using the following rule:

- If the operator is left associative, induce left recursion in its production.

If the operator is right associative, induce right recursion in its production

Example: G: E E+E | E*E | id. Ambiguous grammar converts into unambiguous grammar

Example: Convert the following ambiguous grammar into unambiguous grammar.

G: E → E+E | E*E | E^E | id

E → E+E

E → E*E

E → E^E

E → id

Unambiguous Grammar

Apply Rule 1 & 2

E → E + T | T (+) left associative

T → T * F | F (*) left associative

F → G ^ F | G (^) Right associative

G → id

The priority order: **id** . > ^ . > * . > +

Associativity: + and * are left associative and ^ operator is right associative

Example: Find the precedence and associativity

i) G1: $A \rightarrow A \$ B \mid B$
$B \rightarrow B \# C \mid C$
$C \rightarrow D @ C \mid D$
$D \rightarrow d$

ii) G2: $E \rightarrow E * F \mid F + E \mid F$
$F \rightarrow F - F \mid id$

Consider the grammar defined by the following production rules, with two operators * and +

```
S --> T * P
T --> U | T * U
P --> Q + P | Q
Q --> Id
U --> Id
```

Which one of the following is TRUE?

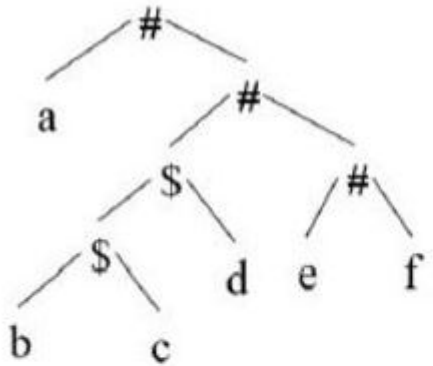
(A) + is left associative, while * is right associative

(B) + is right associative, while * is left associative

(C) Both + and * are right associative

(D) Both + and * are left associative

Consider the following parse tree for the expression a#bcd#e#f, involving two binary operators $



and #. Which one of the following is correct for the given parse tree?

(A) $ has higher precedence and is left associative; # is right associative

(B) # has higher precedence and is left associative; $ is right associative

(C) $ has higher precedence and is left associative; # is left associative

(D) # has higher precedence and is right associative; $ is left associative

Recursive CFG:

Left Factoring:

Parsing Technique:

Brute Force Technique:

Thank You !

First and Follow Function:

First Function:

Follow Function:

Thank You !