BNF :- (Bakus - Naur form)

- Developed by John - Bakus and peter Naur.
- Used for specifying the syntax of the language.
- This notation can be expressed as,

<symbol> : = Exp1 | Exp2 | Exp3 - - - -

Sequence of symbols
(Can be T as well as NT)

<Symbol> → Nonterminal
(written in angle brackets)

Exp → Consisting one or more symbol separated by "|"

reduced by / defined as / can b

: : = → left side must be replaced with the exp of right side.

eg:- <Number> : : = <digit> | <number><digit>

NT NT NT NT

<digit> : : = 0 | 1 | 2 | 3 | - - - - | 9

OR

Terminal

Frenz

The Language of a Grammar :-

- If $G(V, T, P, S)$ is a CFG, the Language of G, denoted $L(G)$, is the set of terminal strings that have derivations from the start symbol. That is,

$$L(G) = \{ w \text{ in } T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

- If a Language L is the Language of some CFG, the L is said to be Context free language (CFL).

Scanned with CamScanner

Closure properties of (CFL)
Context free languages :-

- The context free Language is closed under,
  1. Union
  2. Concatenation
  3. Kleene closure.

1. Union:-

- let $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$ be the two context free ~~G~~ Grammars and L_1 and l_2 be two context free languages.
- We have to show that the union of G_1 and G_2 i.e. $L_1 + L_2$ is also context free
- for that we have to show there exists a CFG G such that G can generate either the strings generated by G_1 or strings generated by G_2

let $G = (V, T, P, S)$ be such CFG

Where,

$$V = \underbrace{V_1}_{\text{NT of } G_1} \cup \underbrace{V_2}_{\text{NT of } G_2} \cup S$$

$$T = T_1 \cup T_2$$

P consists of

$$P_1 \cup P_2 \cup \{S \longrightarrow S_1 | S_2\}$$

S = start symbol

- Here, S is the start symbol of G. The production rule $S \longrightarrow S_1$ tells that G can transit from it's start symbol to start symbol of G_1. i.e. It can use the production role P_1 of G_1 and hence generates the strings generated by G_1. ~~and~~
- Similarly, when it uses the production rule $S \longrightarrow S_2$ tells that G transit from it's start symbol ~~of~~ to start symbol of G_2 i.e. it can use the production roles P_2 of G_2 and generates the strings generated by G_2.
- As a result of which, we can say that G can process either G_1 or G_2. Hence the Class of language is closed under union.

(2) Concatenation :-

- let $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$ be two content free grammars and L_1 & L_2 be ~~two~~ their respective content free language.

We have to show that the concatenation of G_1 and G_2 i.e. $L_1 . L_2$ is also context free.
for that, we have to show there exists a CFG G such that G can generate the strings generated by G_1 and (followed by) strings generated by G_2.

let $G = (V, T, P, S)$ be such CFG,
where,

$V = V_1 \cup V_2 \cup S$

$T = T_1 \cup T_2$

P consists of

$P_1 \cup P_2 \cup \{ S \longrightarrow \cancel{S_1 S_2} \; S_1 . S_2 \}$

S = start symbol.

Here, S is the start symbol of G. the production rule $S \longrightarrow S_1 . S_2$ tells that G can transit from its start symbol to start symbol of G_1 and hence generates the string generated by G_1.
After that G transit from G_1 to G_2 to generate the string generated by G_2.

the class of language is closed under concatenation.

③ Closure :-

⇒ let $G_1 = (V_1, T_1, P_1, S_1)$ be the content free grammar & L_1 be the respective content free language.

- We have to show that the closure of G_1 i.e L_1^* is also content free.
- for that we have to show there exists a grammar G such that G can ~~consist~~ generate the string generated by G_1 any number of times including empty
- let $G = (V, T, P, S)$ be such CFG where

$V = V_1 \cup S$

$T = T_1$

P consists of

$P_1 \cup \{ S \longrightarrow S_1 S_1 | \epsilon \}$

S = start symbol

- Here, S is a start symbol of G, The production rule $S \longrightarrow S_1 S_1$ tells that G transits from start symbol of G to start symbol of G_1 & hence generating the

Frenz

Scanned with CamScanner

string generated by G_1.

- ~~ϕ~~ taking 'ϵ' empty symbol G_1 repeats any number of times to generate the closure of G_1.
- G_1 can transit to G by taking the empty symbol ~~ϕ~~ as well.
- As a result G can produce the closure of G_1. Hence the class of language is closed under closure.

Scanned with CamScanner