For construction of parsing Table we need to find

FIRST

FOLLOW

## FIRST

> In FIRST (1) we see Left Hand Side of the Production Rules.

> FIRST (ϵ) = ϵ
FIRST (terminal) = terminal.

> अगर ϵ आता है तो we have to remove it if it do not come directly. (2)

## F

>In FOLLOW (1) we see Right Hand Side of the produc
n rules.

> FOLLOW में ~~FIRST~~ भी आता है और FOLLOW भी आता है।

> अगर ϵ आता है (2) then we have to remove it.

> FOLLOW of starting symbol always include $

Q-> Find FIRST and FOLLOW for the following Grammar.

$S \longrightarrow AaAb$ — (i)

$S \longrightarrow BbBa$ — (ii)

$A \longrightarrow \in$ — (iii)

$B \longrightarrow \in$ — (iv)

No ambiguity so move further.

Sol:- First of all identify non-terminals

S, A, B

1. FIRST(S) = FIRST(AaAb) ∪ FIRST(BbBa) (i, ii)

= FIRST(A) ∪ FIRST(B)

= $\in$ ∪ $\in$ (iii, iv)

we can't take — we can't take

$\not\in - \not\in$ ∪ FIRST(aAb) ∪ $\not\in - \not\in$ ∪ FIRST(bBa)

a ∪ b

$\{a, b\}$

3. FIRST( B) = { e }. — we have to take as it comes directly.

## FOLLOW

1. FOLLOW (S) = { $ }

2. FOLLOW (A) = FIRST(aAb) U FIRST (b)

a U b

{ a, b }

3. FOLLOW (B) = FIRST ( bBa) U FIRST ( a)

b U a

{a, b}

whose FIRST and FOLLOW of the grammar, productions are given

Start symbol (Z)

$Z \longrightarrow d$ - (i)

$Z \longrightarrow XYZ$ - (ii)

$Y \longrightarrow \epsilon$ - (iii)

$Y \longrightarrow C$ - (iv)

$X \longrightarrow Y$ - (v)

$X \longrightarrow a$ - (vi)

Sol:- Non-terminals

X, Y, Z

Imp

1. FIRST(Z) = FIRST(d) ∪ FIRST(XYZ)

= d ∪ FIRST(X)

= d ∪ FIRST(Y) ∪ FIRST(a)

= d ∪ a ∪ FIRST(Y)

we can't take

d U a U c U ~~ϵ~~ - ~~ϵ~~ U FIRST (Z)

{ a, c, d } U FIRST (Z) — we don't have to go further. as we are finding FIRST(Z) and getting FIRST(Z) in Union.

X { a, c, d } U FIRST (d) U FIRST (XYZ)

no need.

{ a, c, d } U FIRST (Z)

= { a, c, d } Ans.

2. FIRST (Y) = FIRST (C) U ϵ

= { C, ϵ } Ans.

we can't delete ϵ as it comes directly.

3. FIRST (X) = FIRST (Y) U FIRST (a)

Imp

= { C, ϵ } U a

= { a, C, ϵ } Ans.

1. FOLLOW (Z) = { $ } Ans.

as in R.H.S of production rules there is no symbol following Y so it follow X

2. FOLLOW (Y) = FIRST (Z) ∪ FOLLOW (X)

= {a, c, d} ∪ {a, c, d}

= {a, c, d} ∪ {a, c, d}

= {a, c, d} Ans.

3. FOLLOW (X) = FIRST (YZ)

= FIRST (Y)

= {c, ∈} - ∈ ∪ FOLLOW (X) ∪ FIRST (Z)

= {c, ∈} - ∈ ∪ FOLLOW (X) ∪ {a, c, d}

= {a, c, d} Ans.

we can't take ∈ so check for FOLLOW (Y)

grammar whose

$E \rightarrow TE'$

$E' \rightarrow +TE' | \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' | \epsilon$

$F \rightarrow (E) | id$

Sol:- Non terminals are

E, E', T, T', F

**FOLLOW**

**FIRST**

1. FIRST(E) = FIRST(TE')
   = FIRST(T)
   = FIRST(FT')
   = FIRST(F)
   = FIRST((E)) ∪ FIRST(id)
   = ( ∪ id
   = { (, id } Ans.

2. FIRST(E') = FIRST(+TE') ∪ ∈
   = FIRST(+) ∪ ∈
   = { +, ∈ } Ans.

4. FIRST (T') = FIRST (* FT') ∪ ∈

= FIRST (*) ∪ ∈

= { *, ∈ } Ans

5. FIRST (F) = { c, id } Ans.

## FOLLOW

1. FOLLOW (E) = FIRST ( ) ∪ { $ }

= { ), $ } Ans.

2. FOLLOW (E') = FOLLOW (E)

= { ), $ } Ans.

3. FOLLOW (T) = FIRST (E')

= { +, ∈ } − ∈ ∪ FOLLOW(E)

= { +, ), $ } Ans.

we can't take ∈ so use FOLLOW(E')

4. FOLLOW(T') = FOLLOW(T)

= {+, ), $} Ans.

5. FOLLOW(F) = FIRST(T')

= {*, ∈} - ∈ ∪ FOLLOW(T)

= {*, +, ), $} Ans.

# Construction of Predictive Parsing Table

Q-> Construct the parsing Table for the whose productions are given:- Grammar:-

$E \rightarrow TE'$ — (i)

$E' \rightarrow +TE' / \epsilon$ — (ii)

$T \rightarrow FT'$ — (iii)

$T' \rightarrow *FT' / \epsilon$ — (iv)

$F \rightarrow (E) / id.$ — (v)

Sol:- The non-terminals are

E, E', T, T', F

and

FIRST (E) = { (, id } = FIRST (T) = FIRST (F)

FIRST (E') = { +, ∈ }

FIRST (T') = { *, ∈ }

FOLLOW (E) = { ), $ } = FOLLOW (E')

FOLLOW (T) = { +, ), $ } = FOLLOW (T')

FOLLOW (F) = { *, +, ), $ }.

| Non Terminals | id | + | * | ( | ) | $ |
|---|---|---|---|---|---|---|
| E | E → TE' | | | E → TE' | | |
| E' | | E' → +TE' | | | E' → ∈ | E' → ∈ |
| T | T → FT' | | | T → FT' | | |
| T' | | T' → ∈ | T' → *FT' | | T' → ∈ | T' → ∈ |
| F | F → id | | | F → (E) | | |

blank boxes are errors.

1. For E

FIRST(E) = { (, id }  /* both terminals so directly write the production rule of E in table */

2. For E'

FIRST(E') = { +, ∈ }

FOLLOW(E') = { ), $ }  /* one is terminal one is null So check FOLLOW E' */

E' → ∈ वाला rule ) में आएगा।

E' → +TE' वाला rule + में आएगा

4 FIRST (T') = { *, ∈ }

FOLLOW (T') = { +, ), $ }.

5 FIRST (F) = { (, id }.

/* Both are terminals

( comes from F → (E)

id comes from F → id.

For entries of $ attribute

Check where

FIRST ( non Terminal ) = ∈

and

FOLLOW (non Terminal) = $

E', T' so

add E' → ∈

T' → ∈

in $ attribute

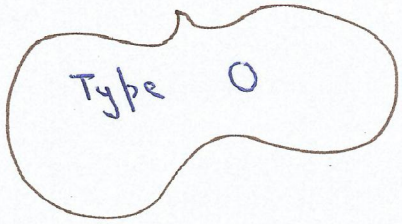
Type 0 ( Recursive Enumerate grammar or Unrestricted grammar )

Type 1 ( CSG or Context Dependent Grammar )

Type 2 ( CFG or Context Independent Grammar or Phrase Structured )

( Nesting of paranthesis

Type 3 ( Regular grammar )




Every production is in the form

$$(V+T)^* \longrightarrow (V+T)^*$$

But remember the basic rule that in every Production rule Atleast one Non Terminal must be there in LHS

e.g

$S \rightarrow aABb$

$aA \rightarrow bBa/b$

$Bb \rightarrow aB \mid Bab/a$

# Type 1

Every Production Rule is in the form

$$(V+T)^* \longrightarrow (V+T)^*$$

+

2 more conditions

(i) $|LHS| \leq |RHS|$

(ii) $RHS \neq \epsilon$

* But remember the basic rule that in every Production rule Atleast one Non terminal must be there in L.H.S

e.g

$S \longrightarrow aSAc \mid abc$

$cA \longrightarrow Ab$

$bA \longrightarrow bb$

Some Imp Points about Type 1

Type 1 grammar is also called as length increasing grammar because The language generated by Type 1 grammar is free from $\epsilon$.

Every Production Rule is in the form

एक variable $\longrightarrow (V+T)^*$

e.g

$S \longrightarrow aSc \mid aAc$

$A \longrightarrow ab \mid \epsilon$

Some Imp points about Type 2

All programming languages are defined by Type 2 grammar

All compilers and parsers understand Type 2 grammar (LR(k) e.t.c)

## Type - 3

Every production Rule is in the form

एक variable $\longrightarrow$ एक या एक से अधिक terminals

या

एक variable $\longrightarrow \epsilon$

या इन दोनों में से एक

एक variable $\longrightarrow$ एक variable. एक या एक से अधिक terminals (Left Linear)

एक variable $\longrightarrow$ एक या एक से अधिक terminals. एक variable (Right Linear)

# Ardens Theorm

The Regular Expression of DFA, NFA is found with the help of Ardens theorm.

**Drawback :-** Ardens theorm cannot be used with ϵ-transition (i.e cannot be used with ϵ-NFA)

**Ardens Theorm :-** eq of each state according to inside arrow to that state

$Y = mY + c$

initial state के साथ ϵ add करना है

only terminals

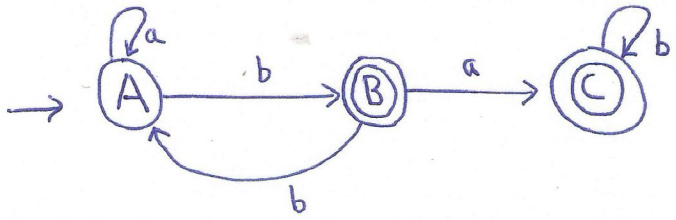
then RE for the state is

$Y = cm^*$

If final states are more than one

R.E for given m/c = R.E one final state + R.E 2nd final state + ....

Q ->




R.E of m/c = $R.E_B + R.E_C$

$A = Aa + Bb + \epsilon$

$B = Ab$

$C = Cb + Ba$

$A = Aa + Bb + \epsilon$

$= Aa + Abb + \epsilon$

$A = A(a+bb) + \epsilon$ ~ S

$R.E = \epsilon (a+bb)^* = (a+bb)^*$

$B = Ab$

$= (a+bb)^* b$

$C = Cb + (a+bb)^* ba$ (r, s)

$R.E = (a+bb)^* bab^*$

$R.E_{m/c} = R.E_B + R.E_C$

$= (a+bb)^* b + (a+bb)^* b ab^*$

$R.E_{m/c} = (a+bb)^* b + (a+bb)^* b ab^*$