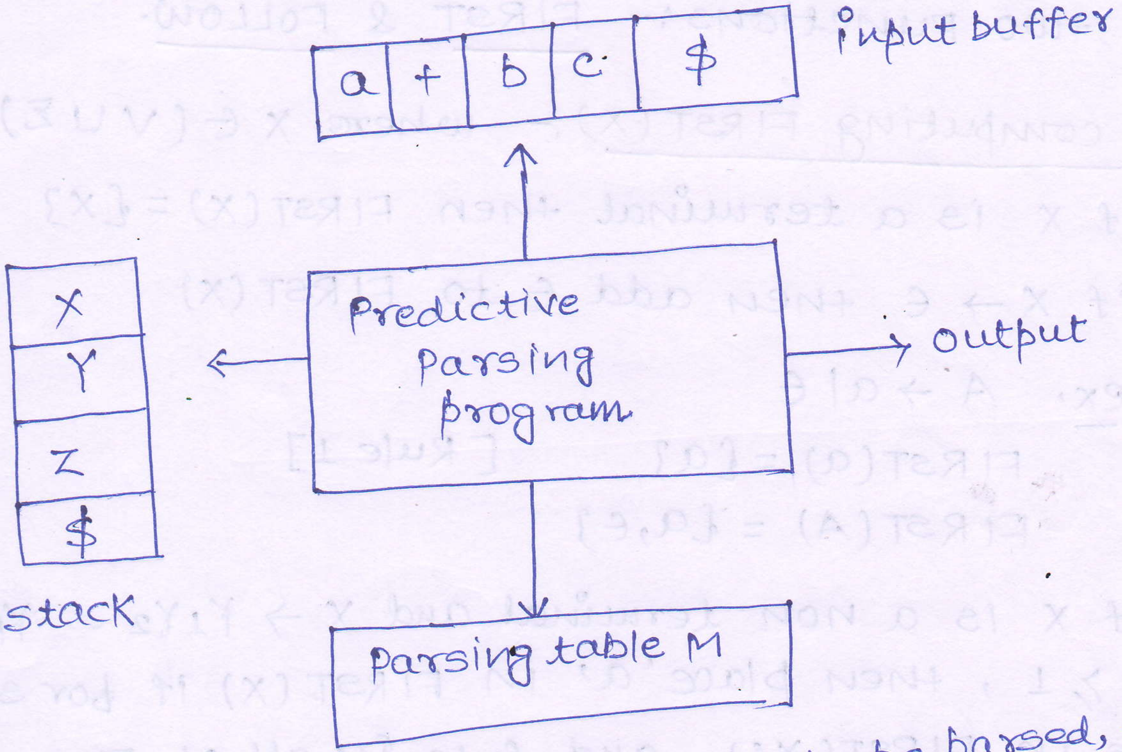
PREDICTIVE PARSING:

Predictive parsing is a special case of recursive descent parsing where no backtracking is required.

→ The predictive parser has an input buffer, stack, a parsing table and an output stream.



input buffer:- It consists of strings to be parsed, followed by $ to indicate the end of the input string.

stack:- It contains a sequence of grammar symbols preceded by $ to indicate the bottom of stack. Initially, the stack contains the start symbol on top of $.

parsing table:- It is a two-dimensional array M [A,a] where 'A' is a non-terminal and 'a' is terminal.

predictive parsing program:- The parser is controlled by a program that consider X, the symbol on top of stack, 'a' the current input symbol. These two symbols determine the parser action. There are three possibilities:-

⟨1⟩ if $X = a = \$$, parser halts and announces successful parsing.

⟨2⟩ if $X = a \neq \$$, the parser pops X off the stack and advances the input pointer to next input symbol.

<3> if X is a non terminal, the program consult entry M[X, a] of the parsing table M. This entry will either be an X production of the grammar or an error entry.

Construction of Predictive Parsing Table→

For construction of Predictive Parsing table, we have to compute two functions :- FIRST & FOLLOW.

Rules for computing FIRST(X) :- where $X \in (V \cup \Sigma)$

Rule 1 :- if X is a terminal then $FIRST(X) = \{X\}$

Rule-2 :- if $X \rightarrow \epsilon$ then add ϵ to FIRST(X)

ex. $A \rightarrow a | \epsilon$

$FIRST(a) = \{a\}$ [Rule 1]

$FIRST(A) = \{a, \epsilon\}$

Rule-3 :- if X is a non terminal and $X \rightarrow Y_1 Y_2 \ldots Y_K$ for some $K \geq 1$, then place 'a' in FIRST(X) if for some i, a is in $FIRST(Y_i)$, and ϵ is in all of $FIRST(Y_1)$, $FIRST(Y_2)$ ......, $FIRST(Y_{i-1})$ i.e. $Y_1, Y_2 \ldots Y_{i-1} \overset{*}{\Rightarrow} \epsilon$. If ϵ is in $FIRST(Y_j)$ for all $j = 1, 2, 3 \ldots K$, then add ϵ to FIRST(X).

$S \rightarrow ABCDE$	$FIRST(S) = \{a, b, c, d, e, \epsilon\}$
$A \rightarrow a \mid \epsilon$	$FIRST(A) = \{a, \epsilon\}$
$B \rightarrow b \mid \epsilon$	$FIRST(B) = \{b, \epsilon\}$
$C \rightarrow c \mid \epsilon$	$FIRST(C) = \{c, \epsilon\}$
$D \rightarrow d \mid \epsilon$	$FIRST(D) = \{d, \epsilon\}$
$E \rightarrow e \mid \epsilon$	$FIRST(E) = \{e, \epsilon\}$

everything in $FIRST(Y_1)$ is in FIRST(X). If Y_1 does not derive ϵ, then we add nothing more to FIRST(X). but $Y_1 \overset{*}{\Rightarrow} \epsilon$ then we add $FIRST(Y_2)$ and so on.

Rules for computing FOLLOW :-

To compute FOLLOW(A) for all non terminals A, apply the following rules until nothing can be added to any FOLLOW set -

Rule-1:- Place \$ in FOLLOW(S), where S is the start symbol and \$ is the input right end marker.

Rule-2: If there is a production $A \rightarrow \alpha B \beta$ then everything in FIRST(β) except ϵ is in FOLLOW(B).

For example→ $S \rightarrow \underset{\alpha}{} \underset{B}{A} \underbrace{BCDE}_{\beta}$

FOLLOW(A) = FIRST(BCDE)

Rule-3:- If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B \beta$ where FIRST(β) contains ϵ, then everything in FOLLOW(A) is in FOLLOW(B).

For example- $E \rightarrow TE'$ FOLLOW(E') = FOLLOW(E)

$A \rightarrow \alpha B$

$S \rightarrow \underbrace{ABC}_{\alpha} \underset{B}{D} \underset{\beta}{E}$ FOLLOW(D) = FIRST(E)

$E \rightarrow e | \epsilon$ = {e, FOLLOW(S)}

FIRST(E) = {e, ϵ}

Example-(1)

$S \rightarrow ABCDE$

$A \rightarrow a | \epsilon$

$B \rightarrow b | \epsilon$

$C \rightarrow c$

$D \rightarrow d | \epsilon$

$E \rightarrow e | \epsilon$

FIRST(S) = {a, b, c}

FIRST(A) = {a, ϵ} [Rule2]

FIRST(B) = {b, ϵ} [Rule 2]

FIRST(C) = {c} [Rule 2]

FIRST(D) = {d, ϵ} [Rule 2]

FIRST(E) = {e, ϵ} [Rule 2]

$S \rightarrow \overset{Y1}{A} \overset{Y2}{B} \overset{Y3}{C} \overset{Y4}{D} \overset{Y5}{E}$ (Rule (3)

FIRST(S) = {a, b, c}

FOLLOW(S) = {\$}

FOLLOW(A) = FIRST(BCDE)
= {b, FIRST(CDE)}
= {b, c}

FOLLOW(B) = FIRST(CDE)
= {c}

FOLLOW(C) = FIRST(DE)
= {d, FIRST(E)}
= {d, e, FOLLOW(S)} = {d, e, \$}

FOLLOW(D) = FIRST(E)
= {e, FOLLOW(S)} = {e, \$}

FOLLOW(E) = FOLLOW(S) = {\$}

(2)

S → Bb | Cd
B → aB | ϵ
C → cC | ϵ

FIRST(S) = {a, b, c, d}
FIRST(B) = {a, ϵ}
FIRST(C) = {c, ϵ}

FOLLOW(S) = {\$}
FOLLOW(B) = {b}
FOLLOW(C) = {d}

B → aB
FOLLOW(B) = FOLLOW(B)

S → Bb
FOLLOW(B) = FIRST(b)
= {b}

(3)

E → TE'
E' → +TE' | ϵ
T → FT'
T' → *FT' | ϵ
F → (E) | id

FIRST(E) = FIRST(T) = {id, (}
FIRST(E') = {+, ϵ}
FIRST(T) = FIRST(F) = {id, (}
FIRST(T') = {*, ϵ}
FIRST(F) = {id, (}

F → (E)
FOLLOW(E) = {) }

E is also start symbol so
FOLLOW(E) = {\$,)}

FOLLOW(E') = {\$,)}
FOLLOW(T) = {+, \$,)}
FOLLOW(T') = {+, \$,)}
FOLLOW(F) = {*, +, \$,)}