

FIRST AND FOLLOW

Rules for FIRST SET

- If α is a terminal 'a' then $\text{First}(\alpha)$ is $\{a\}$.
 - Eg. If $A \rightarrow a$, then $\text{First}(A) = \{a\}$.
- If α is a non terminal 'X' & $X \rightarrow A$ & $A \rightarrow a$, then $\text{First}(X) = \{a\}$.
- If α is a non terminal 'X', & $X \rightarrow \epsilon$, then $\text{First}(X) = \{\epsilon\}$.
 - Eg. If $A \rightarrow \epsilon$, then $\text{First}(A) = \{\epsilon\}$.
- If $X \rightarrow Y_1, Y_2, Y_3, \dots, Y_k$ then $\text{First}(X) = \text{First}(Y_1)$. If $\text{First}(Y_1)$ has ϵ then X has $\text{First}(Y_2)$ and so on.

Note – First sets can contain 'ε'.

Rules for FOLLOW SET

1. Whenever you want to find follow of a symbol, find the symbol in R.H.S.
2. If symbol is start symbol, add \$ to follow set.
3. If symbol is followed by a terminal, straightway add the terminal to the follow set.
4. If the symbol is followed by non-terminal, find the first of the non-terminal.
 - 4.1 – If the first set contains ϵ , exclude ϵ from follow sets & replace it in the production.
 - 4.2 – If even after replacement by ϵ , it is followed by terminal or non terminal, repeat the process.
5. If the symbol is last one in the production, then add follow of head into follow set.

Note – Follow set do not contain ' ϵ '

Points to remember while constructing Parse Table

- ✓ Make a empty table in which rows are labelled with the non-terminals and columns with the terminals of the grammar.
- ✓ \$ is explicitly added in every parse table.
- ✓ Never add Epsilon in the parse table.

Construction of Predictive Parse Table

Algorithm : Construction of a predictive parsing table

Input: Grammar G

Output: Parsing table M

Method:

- ❶ For each production of the form $A \rightarrow \alpha$ of the grammar, do steps 2 and 3
- ❷ For each terminal symbol '**a**' in $\text{FIRST}(\alpha)$, add $A \rightarrow \alpha$ to $M[A, a]$
- ❸ If ϵ is in $\text{FIRST}(\alpha)$ add $A \rightarrow \alpha$ to $M[A, b]$ for each terminal b in $\text{FOLLOW}(A)$. If ϵ is in $\text{FIRST}(\alpha)$ and $\$$ is in $\text{FOLLOW}(A)$, add $A \rightarrow \alpha$ to $M[A, \$]$
- ❹ Make each undefined entry of M be **error**.

Steps needs to be followed for constructing a parse table and to parse a given input string.

- ✓ Eliminate Left Recursion (if any) in the grammar G.
- ✓ Perform left factoring on G.
- ✓ Find FIRST and FOLLOW on the symbols in G.
- ✓ Construct predictive parse table.
- ✓ Check if the input string is accepted by parser using the parse table entries.

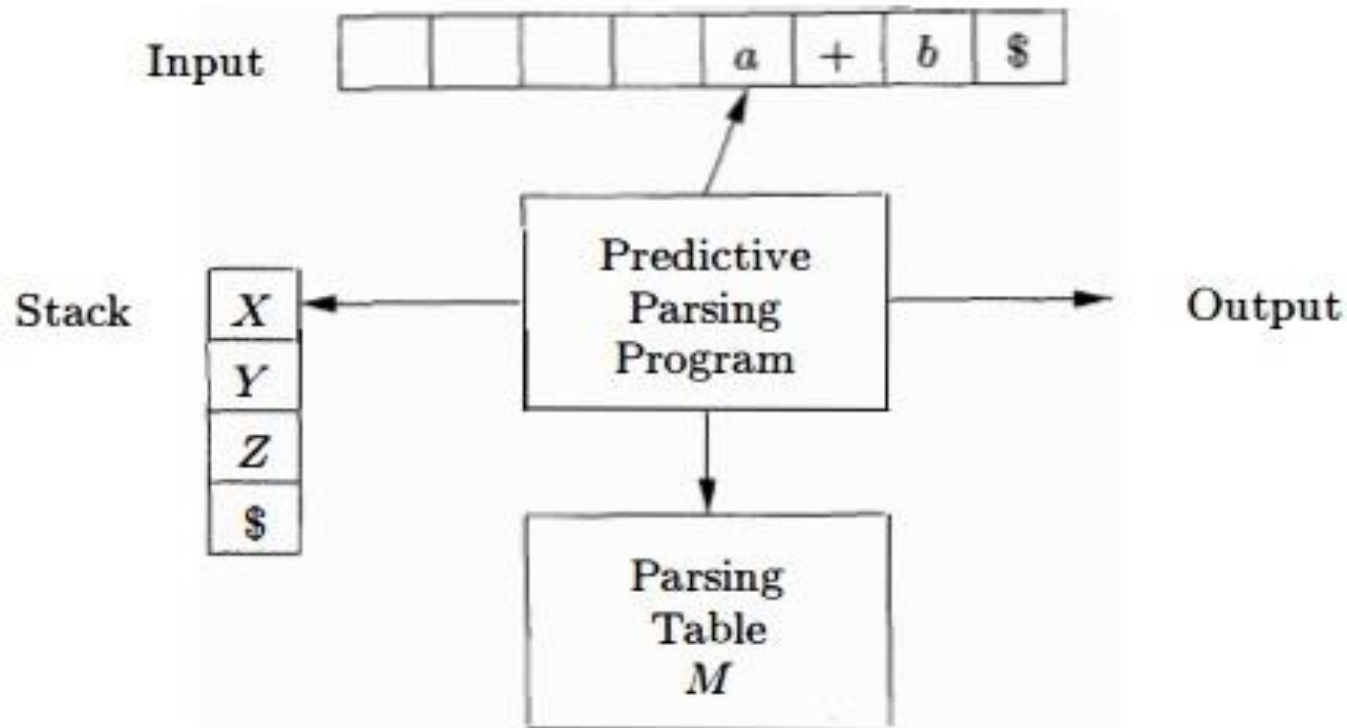
Non –Recursive Predictive Parsing

- It is built maintaining a stack.
- The table driven parser has an
 - Input buffer
 - Stack containing sequence of grammar symbols
 - Parsing tables
 - Output stream
- Input buffer:
 - Has the string to be parsed.
 - End of the string is marked with a \$.
- Stack
 - \$ marks the bottom of the stack.
 - Initially it has start symbol of the grammar on the top.

Table Driven Predictive parsing Algorithm

INPUT : A string w and a parsing table M for grammar G .

OUTPUT : If w is in $L(G)$, a leftmost derivation of w ; otherwise, an error indication.



Predictive parsing Algorithm

```
set ip to point the first symbol a of w
set X to the top stack symbol
while(X!=$)
{
    if (X is a) then pop the stack and advance ip
    elseif (X is a terminal) error();
    elseif (M[X,a] is an error entry) error();
    elseif (M[X,a]=X->Y1,Y2,...,Yk)
    {
        output the production X->Y1,Y2,...,Yk
        pop the stack
        push Yk,Yk-1,...,Y1 onto the stack with Y1 on top
    }
    set X to the top stack symbol
}
```

