TERM PAPER ON

PARSING

COURSE CODE: CAP632

SECTION: D1809

COURSES NAME:

FORMAL LANGUAGES AND AUTOMATION THEORY

SUBMITED BY:

11607193 Shrikant Sharma

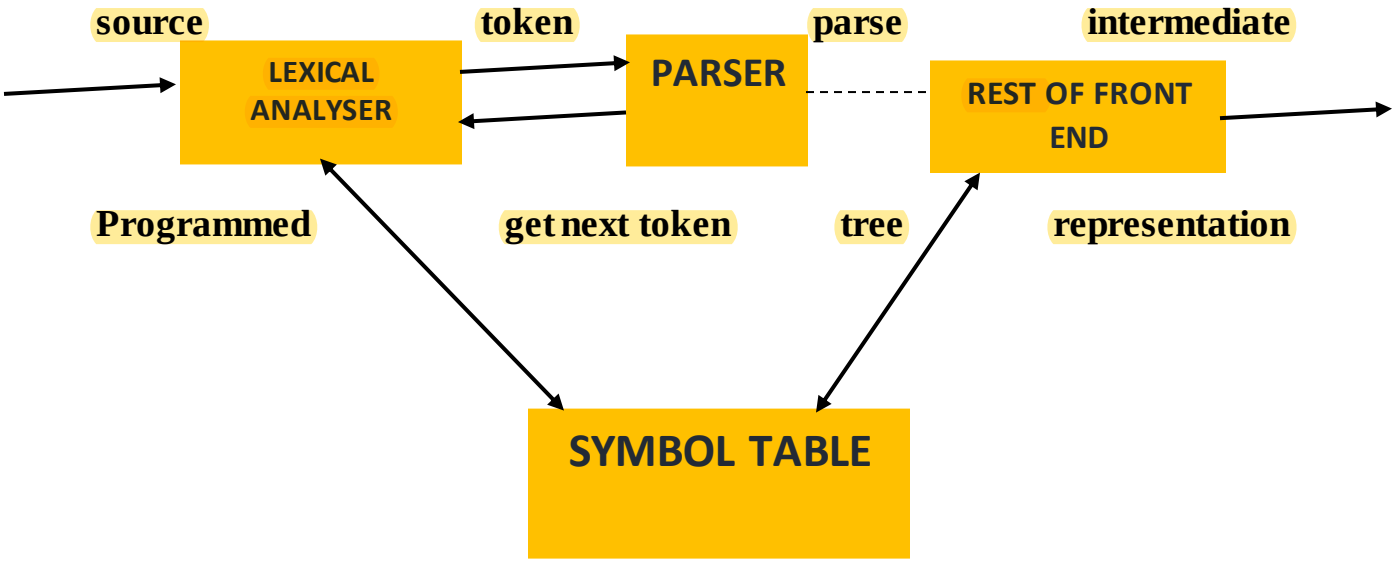
SUBMITED TO:

Miss Pallvi

What is Parser?

- A Parser is a compiler or interpreter component that breaks the data into smaller elements for easy translation into another language.
- It takes input form of sequence of tokens or program instructions and build the data structure in the form of parse tree.

What is the Role of Parser?



- In the compiler model, the parser obtains a string of token from the lexical analyser, and verifies the string can be generated by the grammar for the source language.
- The parser returns any syntax error for the source language.
- It collects sufficient number of tokens and builds a parse tree.

What is Parsing?

- Parsing is used to derived a string using the production rules of grammar.
- It is used to check the acceptability of string.

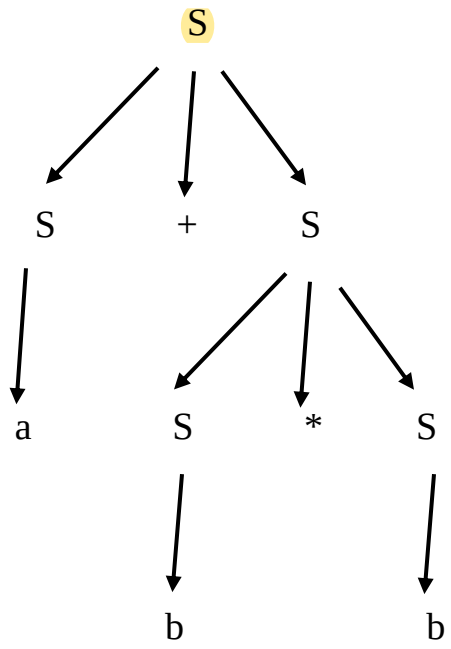
Example of Parsing

Problem: Consider the grammar

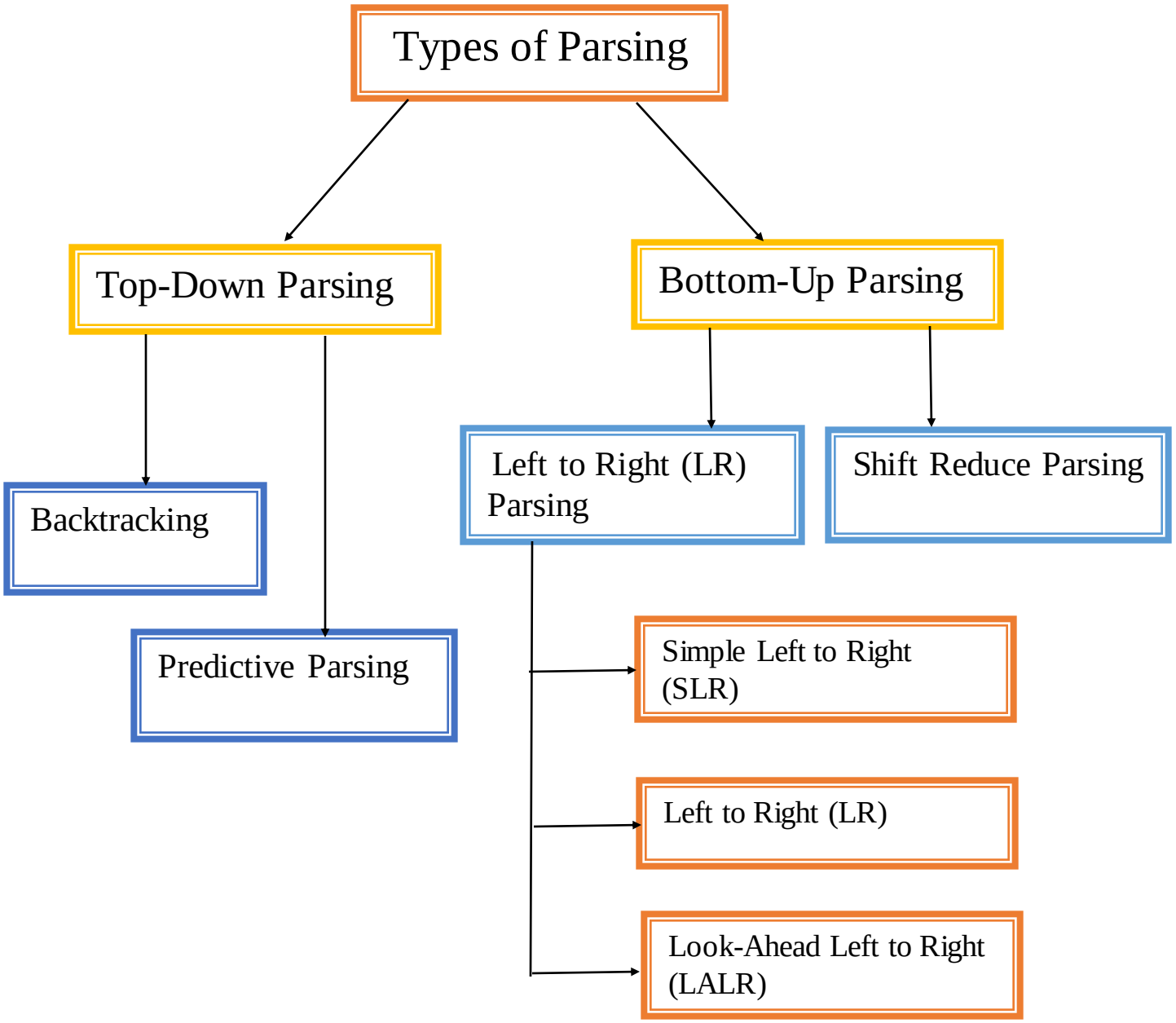
S=S+S | S*S | a | b

Construct the derivation tree or parse tree for the string (w = a + b * b)

Solution:



Types of Parsing



Types of Parsing

There are two types of parsing:

(i) Top-Down Parsing
(ii) Bottom-Up Parsing

Top-Down Parsing

- Top-Down Parsing starts from the top with the start symbol and derives a string using parse tree.
- It may Backtracking.

Design of Top-Down Parsing

For top-down parsing, a PDA has the following four types of transitions −

- Pop the non-terminal on the left-hand side of the production at the top of the stack and push its right-hand side string.
- If the top symbol of the stack matches with the input symbol being read, pop it.
- Push the start symbol 'S' into the stack.
- If the input string is fully read and the stack is empty, go to the final state 'F

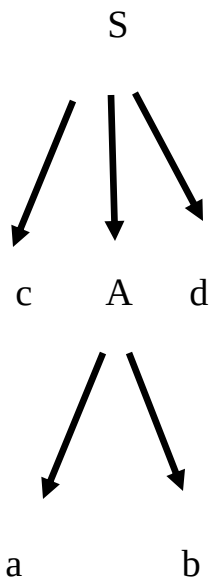
Example of Top-Down Parsing

Example 1: Let us consider the following grammar.

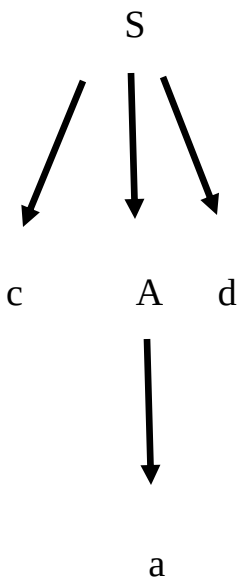
S -> cAd

A -> ab | a

Let input string w = cad



(a)



(b)

Let written in programming language:

Procedure S

Procedure **S ()**,

Begin

If input symbol= **'c'** then

Begin

ADVANCE ();

If **A ()** then

If input symbol= **'d'** then

Begin ADVANCE (),

Return true

End

End

Return false

End

Procedure A ()

```
Procedure A (),
  Begin
   isave=input-pointer;
      If input symbol= 'a' then
       Begin
            ADVANCE (),
               If input symbol= 'b' then
                  Begin   ADVANCE (),
                     Return true
                        End
                           End
```

** if **return true** will not be terminated.

```
input-pointer=isave;
      If input symbol= 'a' then
       Begin
            ADVANCE (),
                  Return true
                     End
                       Else
                         Return false
                            End
```

Example 2: let consider these equation

E -> E+E

E -> E*E

E -> a

E -> b

Input strings: (a*b) + (a*b)

**** This parsing technique uses Left Most Derivation.**

Solution:

E -> E+E

E -> E*E + E

E -> a*E+E

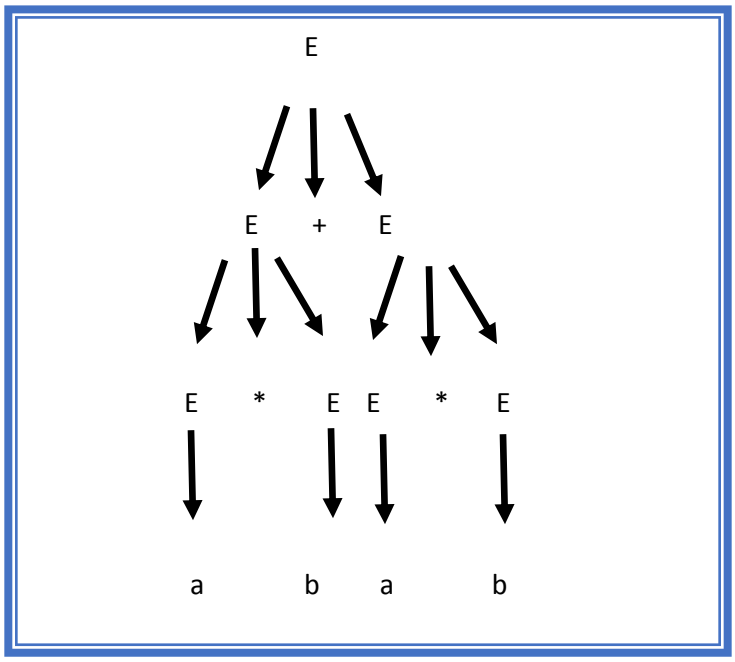
E -> a*b + E

E -> a*b + E+E

E -> a*b + E*E

E -> a*b + a* E

E -> a*b + a*b



Problem of Top-Down Parsing

(i) Backtracking

- Backtracking is a technique in which for expansion of non-terminal symbol we choose one alternative and if some mismatch occurs then we try to other alternative if any.

(ii) Left Recursion

- Left Recursion is removed if the parser performs top-down parsing.

(iii) Left Factoring

- ❖ Left Factoring is removing the common left factor that appears in two productions of the same non-terminal. It is done to avoid the backtracking by the parser.

Bottom-Up Parsing

- Bottom-Up Parsing starts from the bottom with the string and comes to the start symbol using parse tree.
- It uses stack to store both state and sentential forms.

Design of a Bottom-Up Parser

For bottom-up parsing, a PDA has the following four types of transitions −

- Push the current input symbol into the stack.
- Replace the right-hand side of a production at the top of the stack with its left-hand side.
- If the top of the stack element matches with the current input symbol, pop it.
- If the input string is fully read and only if the start symbol 'S' remains in the stack, pop it and go to the final state 'F'.

Example of Bottom-Up Parsing

let consider these equation

E -> E+E

E -> E*E

E -> a

E -> b

Input strings: (a*b) + (a*b)

**** This parsing technique uses Right Most Derivation.**

Solution:

E -> E+E

E -> E + E*E

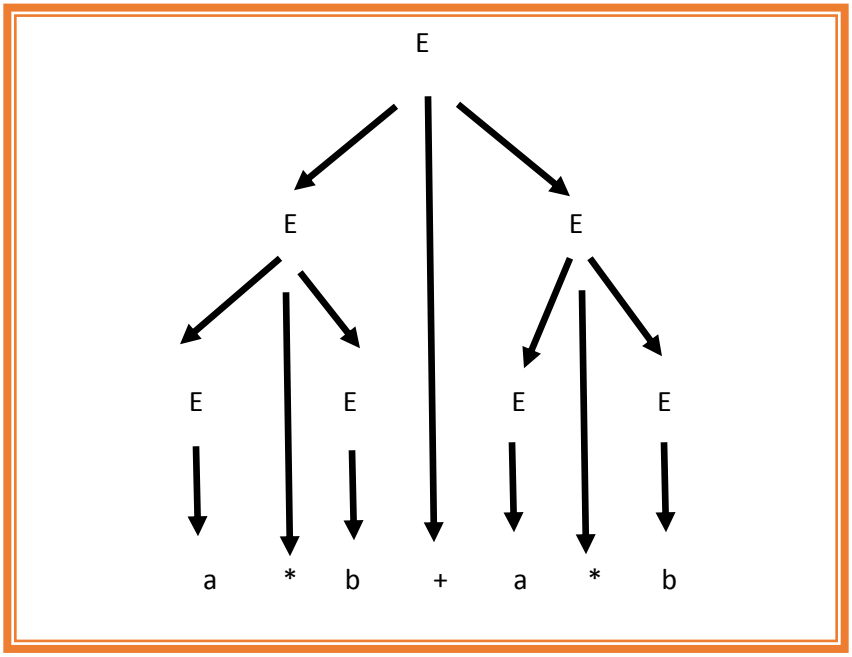
E -> E + E * b

E -> E + a * b

E -> E*E + a*b

E -> E * b + a*b

E -> a*b + a*b



References

https://www.slideshare.net/khush_boo31/parsing-67077365

https://www.youtube.com/watch?v=WYb-Iblk7J0

https://www.youtube.com/watch?v=42nqWoHacxg

https://www.youtube.com/watch?v=7LwxK2B_H3k