CSE-3315: COMPILER DESIGN AND CONSTRUCTION

LECTURE –8

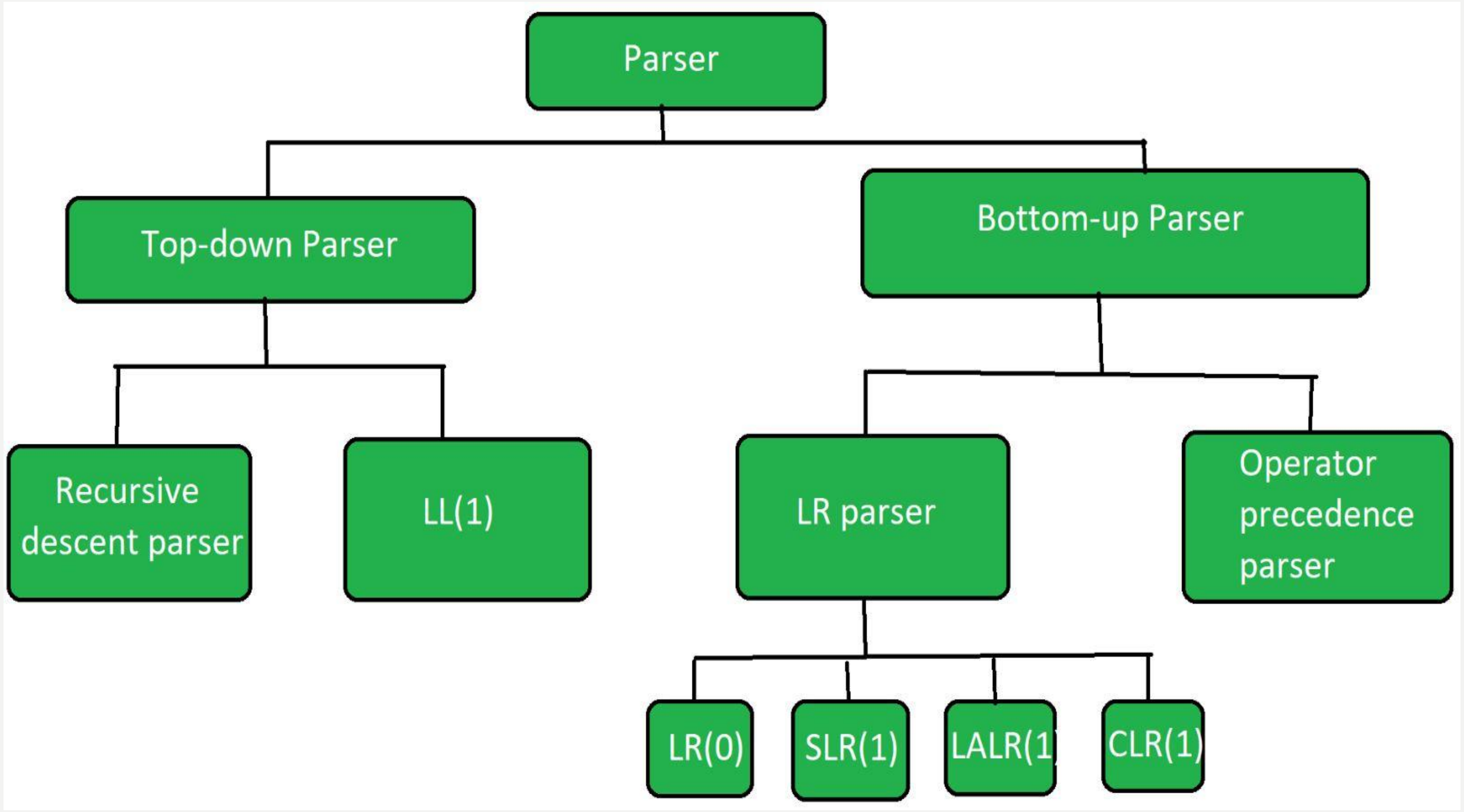
Zumana Islam Mou
Adjunct Lecturer, CSE, Leading University

CHAPTER 4

SYNTAX ANALYSIS

COMPILERS: *PRINCIPLES, TECHNIQUES, & TOOLS*
ALFRED V. AHO, MONICA S. LAM, RAVI SETHI, JEFFREY D. ULLMAN
SECOND EDITION

TYPES OF PARSER

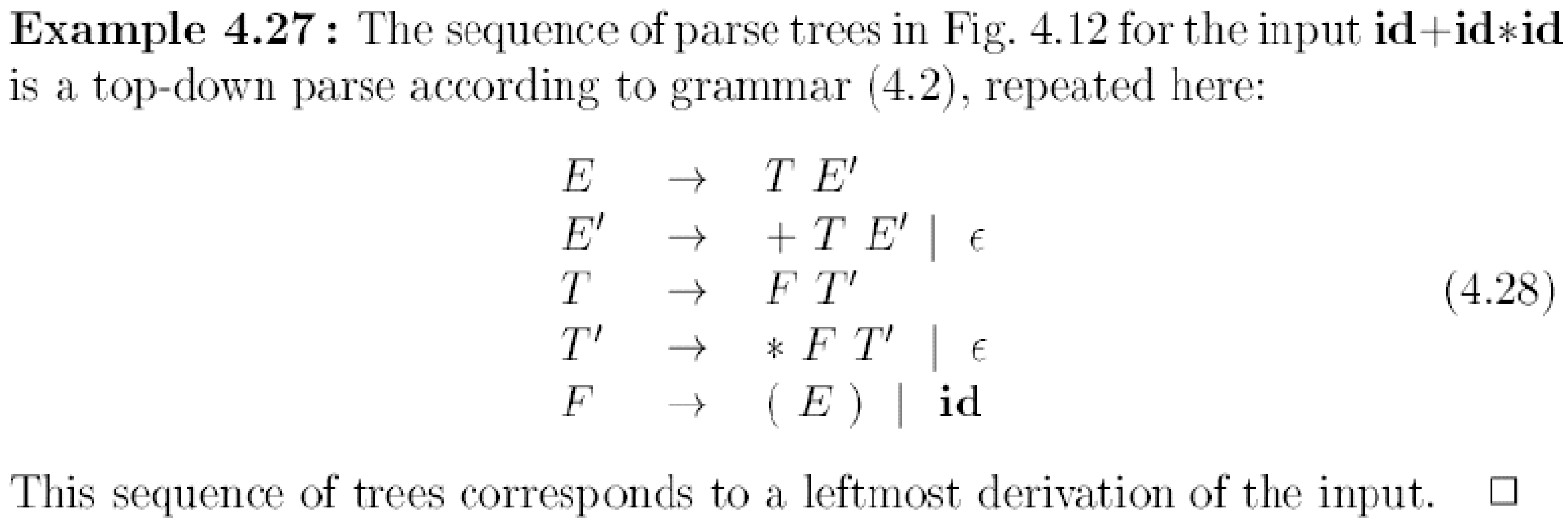


TOP-DOWN PARSING

- Top-down parsing can be viewed as the problem of constructing a parse tree for the input string, starting from the root and creating the nodes of the parse tree in preorder.

- Equivalently, top-down parsing can be viewed as finding a leftmost derivation for an input string.

Top-Down Parsing



Example 4.27 : The sequence of parse trees in Fig. 4.12 for the input **id**+**id**$*$**id** is a top-down parse according to grammar (4.2), repeated here:

$$\begin{array}{lcl} E & \rightarrow & T\ E' \\ E' & \rightarrow & +\ T\ E' \mid \epsilon \\ T & \rightarrow & F\ T' \\ T' & \rightarrow & *\ F\ T' \mid \epsilon \\ F & \rightarrow & (\ E\ ) \mid \mathbf{id} \end{array} \tag{4.28}$$

This sequence of trees corresponds to a leftmost derivation of the input. □

Top-Down Parsing

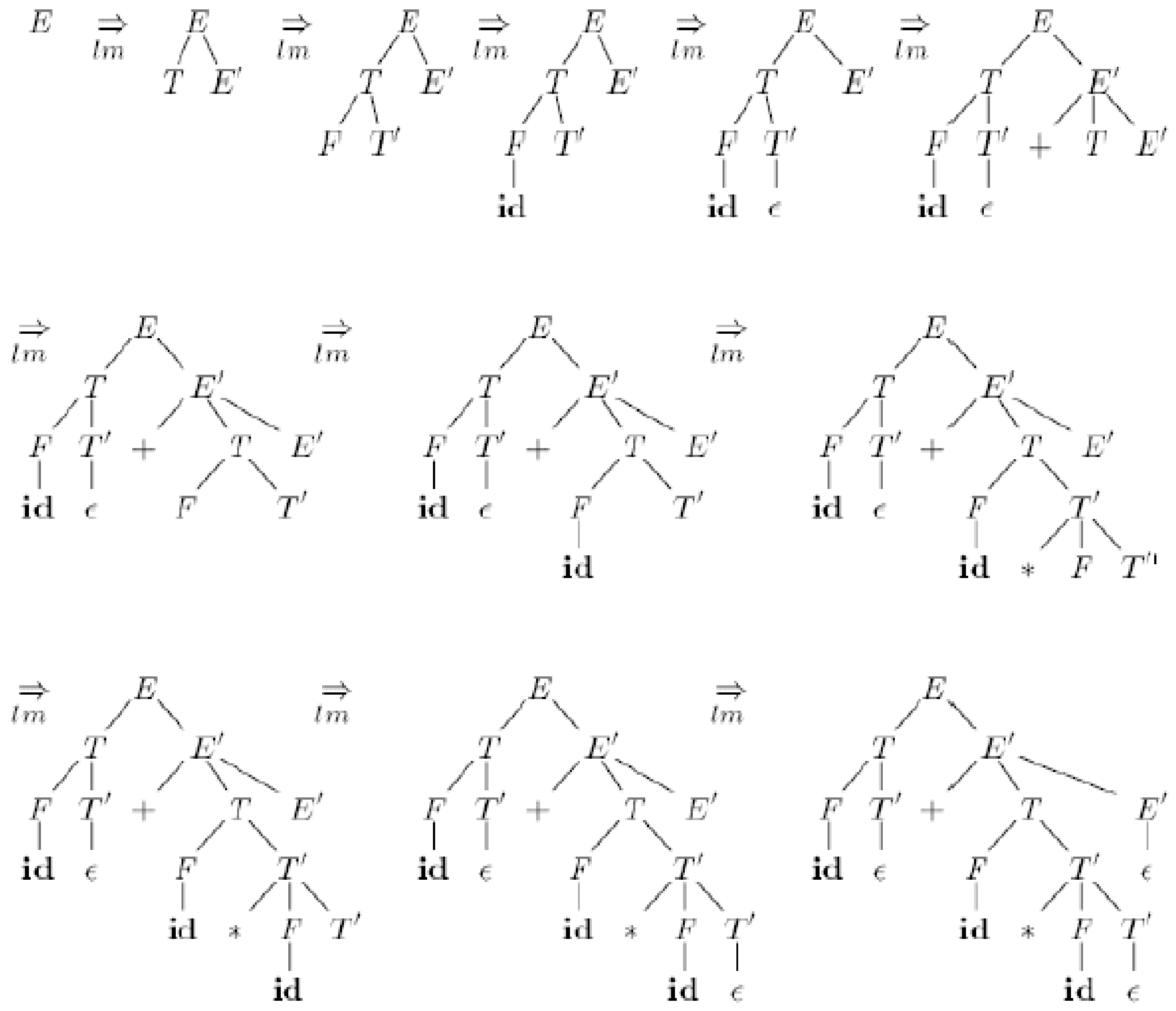


Figure 4.12: Top-down parse for **id** + **id** * **id**

TOP-DOWN PARSING

- At each step of a top-down parser, the key problem is that of determining the production to be applied for a nonterminal, say A.
- Once an A-production is chosen, the rest of the parsing process consists of “matching" the terminal symbols in the production body with the input string.

RECURSIVE-DESCENT PARSING

- Recursive Descent Parser is a top-down method of syntax analysis in which a set of **recursive procedures** is used to process input.
- Execution begins with the procedure for the start symbol, which halts and announces success if its procedure body scans the entire input string.
- It is also known as the **Brute force parser** or the **backtracking parser**.
- It basically generates the parse tree by using brute force and backtracking.

Recursive-Descent Parsing

Example 4.29 : Consider the grammar

$$\begin{aligned} S &\rightarrow c\,A\,d \\ A &\rightarrow a\,b \mid a \end{aligned}$$

To construct a parse tree top-down for the input string $w = cad$.

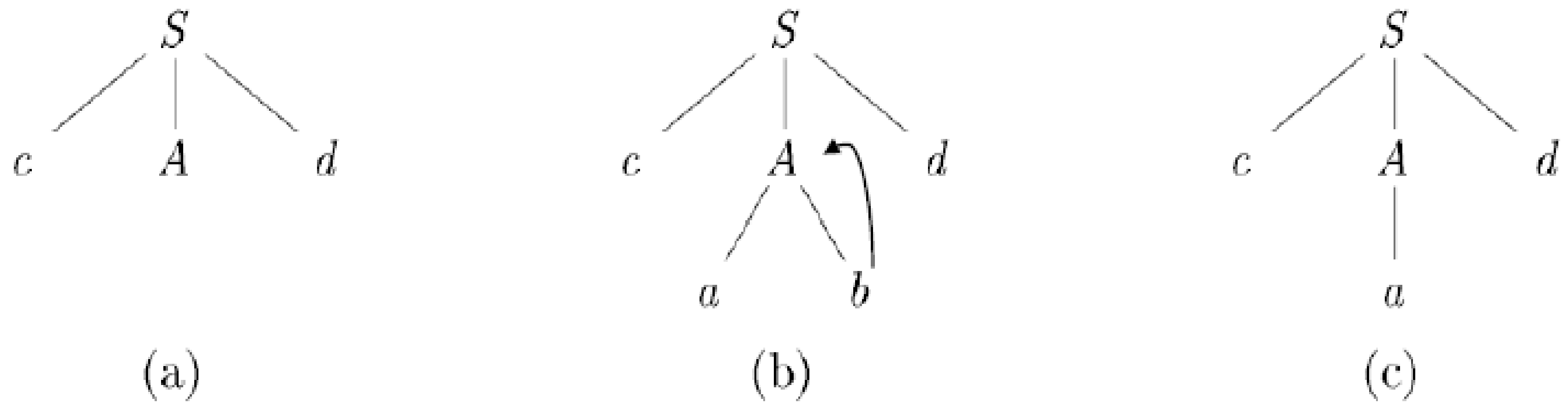


Figure 4.14: Steps in a top-down parse

RECURSIVE-DESCENT PARSING

- A LEFT-RECURSIVE GRAMMAR CAN CAUSE A RECURSIVE-DESCENT PARSER, EVEN ONE WITH BACKTRACKING, TO GO INTO AN INFINITE LOOP.
- THAT IS, WHEN WE TRY TO EXPAND A NONTERMINAL A, WE MAY EVENTUALLY FIND OURSELVES AGAIN TRYING TO EXPAND A WITHOUT HAVING CONSUMED ANY INPUT.

NON-RECURSIVE PREDICTIVE PARSING

- It is also known as LL(1) parser or predictive parser or without-backtracking parser.
- It uses a parsing table to generate the parse tree instead of backtracking.
- It has the capability to predict which production is to be used to replace the input string.
- Predictive parsing uses a stack and a parsing table to parse input and generate a parse tree.

PREDICTIVE PARSING

- The goal of predictive parsing is to construct a top-down parser that never backtracks.
- To do so, we must transform grammar in two ways:

1. eliminate left recursion
2. perform left factoring

- These rules eliminate the most common causes for backtracking.