Compiler Design

Fatemeh Deldar

Isfahan University of Technology

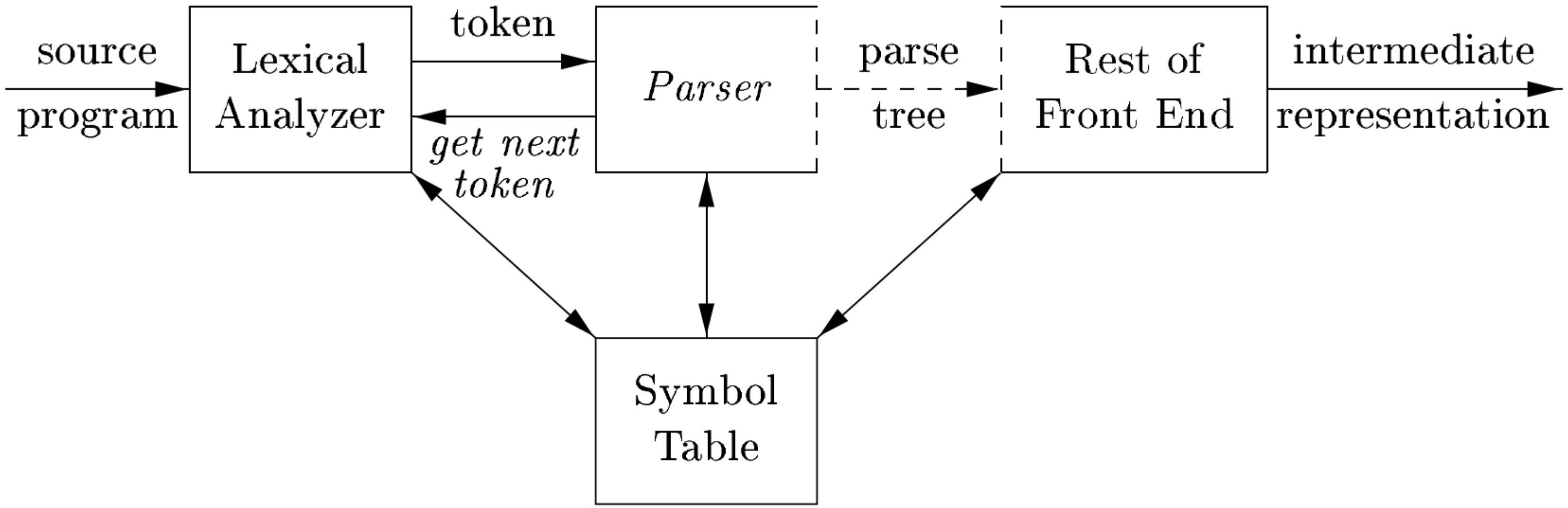
1403-1404

Syntax Analysis

Introduction

- Every programming language has precise rules that prescribe the syntactic structure of well-formed programs
- The syntax of programming language constructs can be specified by **context-free grammars**
- The parser **obtains a string of tokens from the lexical analyzer** and **verifies that the string of token names can be generated by the grammar for the source language**
- The parser constructs a parse tree and passes it to the rest of the compiler for further processing

Introduction



Introduction

- The methods commonly used in compilers for parser can be classified as:
  - **Top-down**
    - Top-down methods build parse trees from the top (root) to the bottom (leaves)
  - **Bottom-up**
    - Bottom-up methods start from the leaves and work their way up to the root
- The input to the parser is scanned from left to right, one symbol at a time

Context-Free Grammars

- **A context-free grammar consists of:**
  - **Terminals**
    - Terminals are the basic symbols from which strings are formed
  - **Nonterminals**
    - Nonterminals are syntactic variables that denote sets of strings
  - **Start symbol**
    - One nonterminal is distinguished as the start symbol
  - **Productions**
    - The productions of a grammar specify the manner in which the terminals and nonterminals can be combined to form strings
    - **A production consists of:**
      - A nonterminal called the head or left side of the production
      - The symbol $\rightarrow$
      - A body or right side consisting of zero or more terminals and nonterminals

Context-Free Grammars

- **Example**
  - Terminal: id + - * / ( )
  - Nonterminals: expression, term, factor

$$
\begin{array}{rcl}
\mathit{expression} & \rightarrow & \mathit{expression} + \mathit{term} \\
\mathit{expression} & \rightarrow & \mathit{expression} - \mathit{term} \\
\mathit{expression} & \rightarrow & \mathit{term} \\
\mathit{term} & \rightarrow & \mathit{term} * \mathit{factor} \\
\mathit{term} & \rightarrow & \mathit{term} \,/\, \mathit{factor} \\
\mathit{term} & \rightarrow & \mathit{factor} \\
\mathit{factor} & \rightarrow & (\ \mathit{expression}\ ) \\
\mathit{factor} & \rightarrow & \mathbf{id}
\end{array}
$$

- **Example**

$$
\begin{array}{lcl}
E & \rightarrow & E + T \mid E - T \mid T \\
T & \rightarrow & T * F \mid T \,/\, F \mid F \\
F & \rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
$$

Derivations

- **Example** $E \rightarrow E + E \mid E * E \mid - E \mid (E) \mid \textbf{id}$
  - A derivation of -(**id**) from E is: $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(\textbf{id})$
- If $S \overset{*}{\Rightarrow} \alpha$, where S is the start symbol of a grammar G, we say that α is a **sentential form** of G
  - A sentential form may contain both terminals and nonterminals
  - The strings E, -E, -(E), -(id + id) are all sentential forms
- A **sentence** of G is a sentential form with no nonterminals
- **The language generated by a grammar is its set of sentences**
- **Example**

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E) \Rightarrow -(\textbf{id} + E) \Rightarrow -(\textbf{id} + \textbf{id})$$

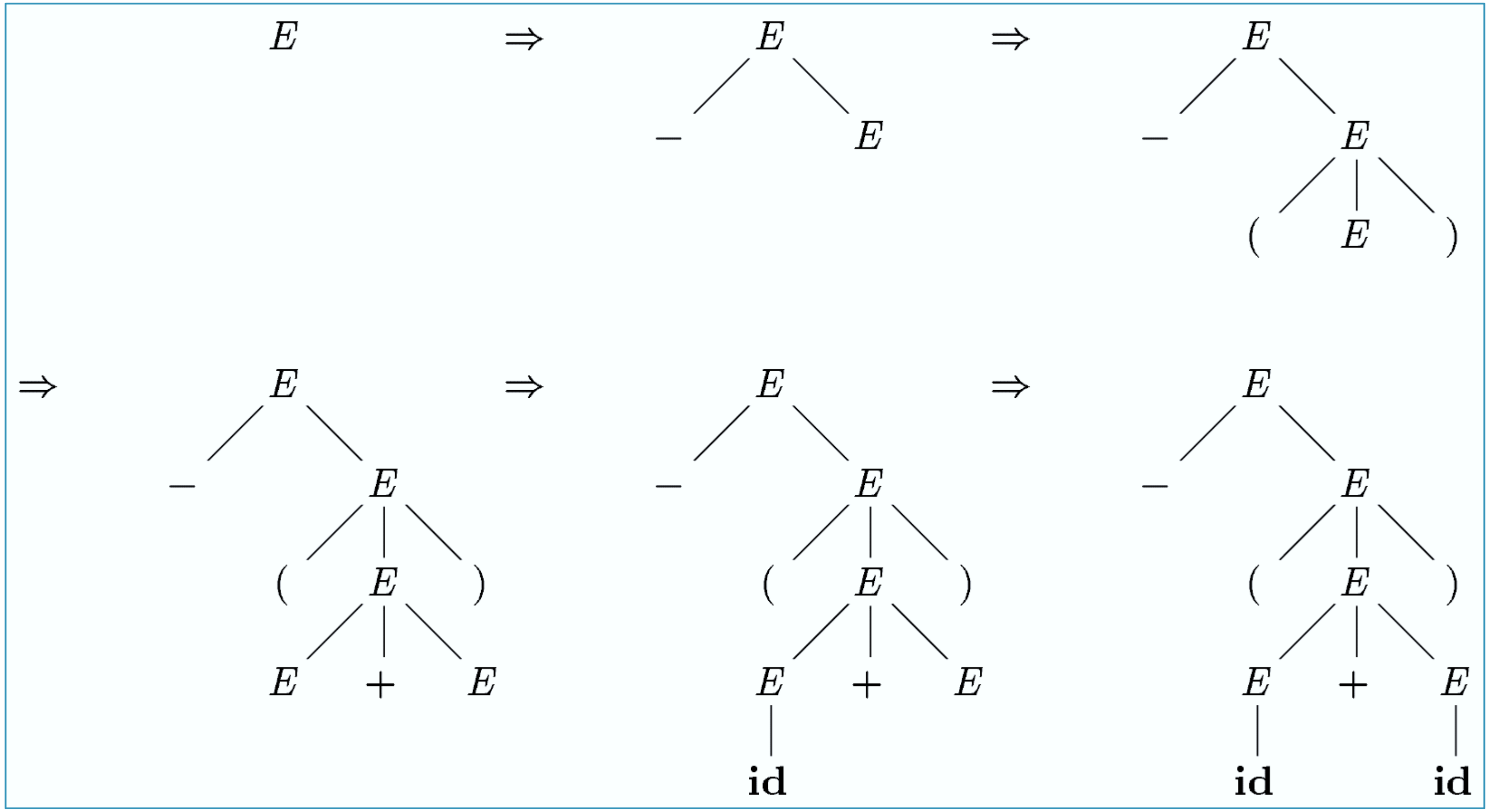
Derivations

- **Leftmost derivations** $\boxed{\alpha \underset{lm}{\Rightarrow} \beta}$
  - The leftmost nonterminal in each sentential is always chosen
- **Rightmost derivations** $\boxed{\alpha \underset{rm}{\Rightarrow} \beta}$
  - The rightmost nonterminal in each sentential is always chosen
- **Example**

$$\boxed{E \underset{lm}{\Rightarrow} -E \underset{lm}{\Rightarrow} -(E) \underset{lm}{\Rightarrow} -(E+E) \underset{lm}{\Rightarrow} -(\mathbf{id}+E) \underset{lm}{\Rightarrow} -(\mathbf{id}+\mathbf{id})}$$

Parse Trees

- A parse tree is a graphical representation of a derivation that filters out the order in which productions are applied to replace nonterminals

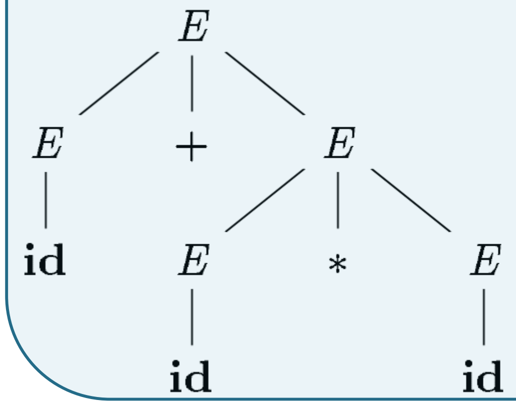


Ambiguity

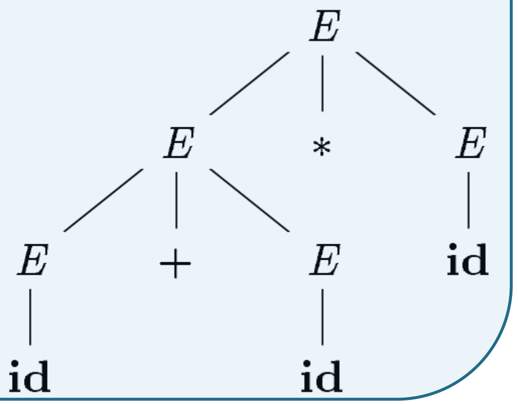
- A grammar that produces more than one parse tree for some sentence is said to be **ambiguous**
  - An ambiguous grammar is one that produces more than one leftmost derivation or more than one rightmost derivation for the same sentence

$$E \rightarrow E + E \mid E * E \mid - E \mid (E) \mid \mathbf{id}$$

$$\begin{aligned} E &\Rightarrow E + E \\ &\Rightarrow \mathbf{id} + E \\ &\Rightarrow \mathbf{id} + E * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id} \end{aligned}$$



$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow E + E * E \\ &\Rightarrow \mathbf{id} + E * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id} \end{aligned}$$



Ambiguity

- مثال: یک گرامر غیرمبهم معادل برای گرامر زیر بنویسید.

- $E \rightarrow E + E \mid E - E \mid E * E \mid E \,/\, E \mid E \wedge E \mid -E \mid (E) \mid \mathrm{id}$

- $E \rightarrow E + T \mid E - T \mid T$
- $T \rightarrow T * F \mid T \,/\, F \mid F$
- $F \rightarrow P \wedge F \mid P$
- $P \rightarrow -P \mid M$
- $M \rightarrow (E) \mid \mathrm{id}$

Verifying the Language Generated by a Grammar

- A proof that a grammar G generates a language L has two parts:
  - Show that every string generated by G is in L
  - Show that every string in L can indeed be generated by G
- **Example**
  - **The following grammar generates all strings of balanced parentheses**

$$S \to (\; S \;) \; S \;\; | \;\; \epsilon$$

Context-Free Grammars Versus Regular Expressions

- **Every regular language is a context-free language, but not vice-versa**
- **Example**
  - Construct a context-free grammar from an NFA

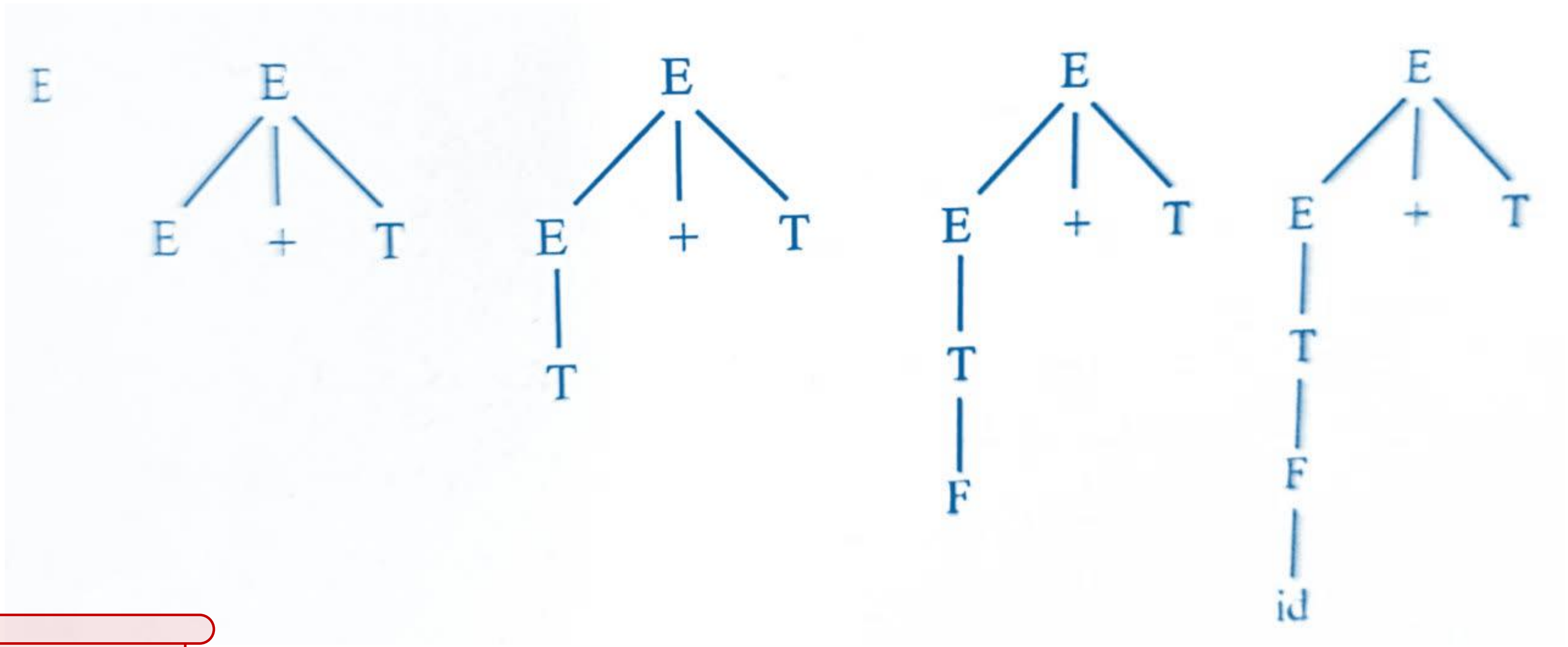
$$(a|b)^*abb$$

$$\begin{array}{lcl} A_0 & \rightarrow & aA_0 \mid bA_0 \mid aA_1 \\ A_1 & \rightarrow & bA_2 \\ A_2 & \rightarrow & bA_3 \\ A_3 & \rightarrow & \epsilon \end{array}$$

- The language $L = \{a^n b^n \mid n \geq 1\}$ is context-free but not regular
- ***Finite automata cannot count***

سمت چپ‌ترین اشتقاق

Top-Down Parsing

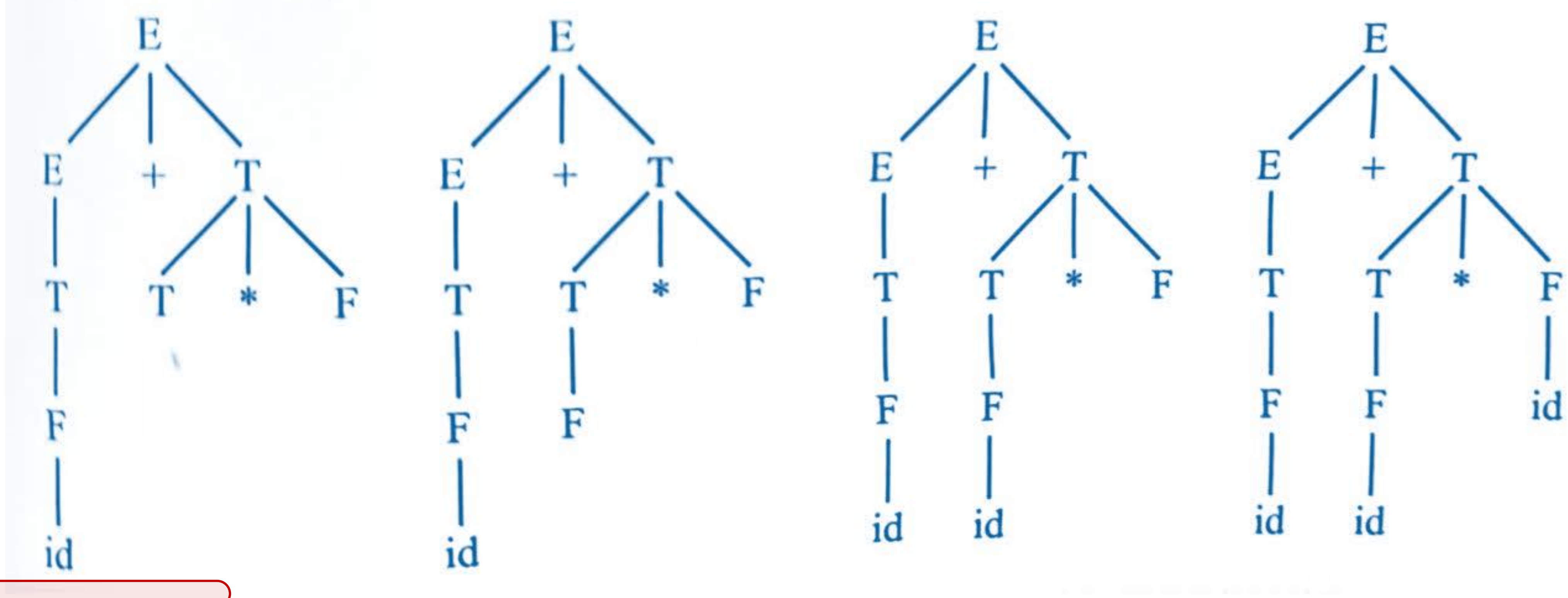


Example:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \mathrm{id}$$

سمت چپ‌ترین اشتقاق

Top-Down Parsing

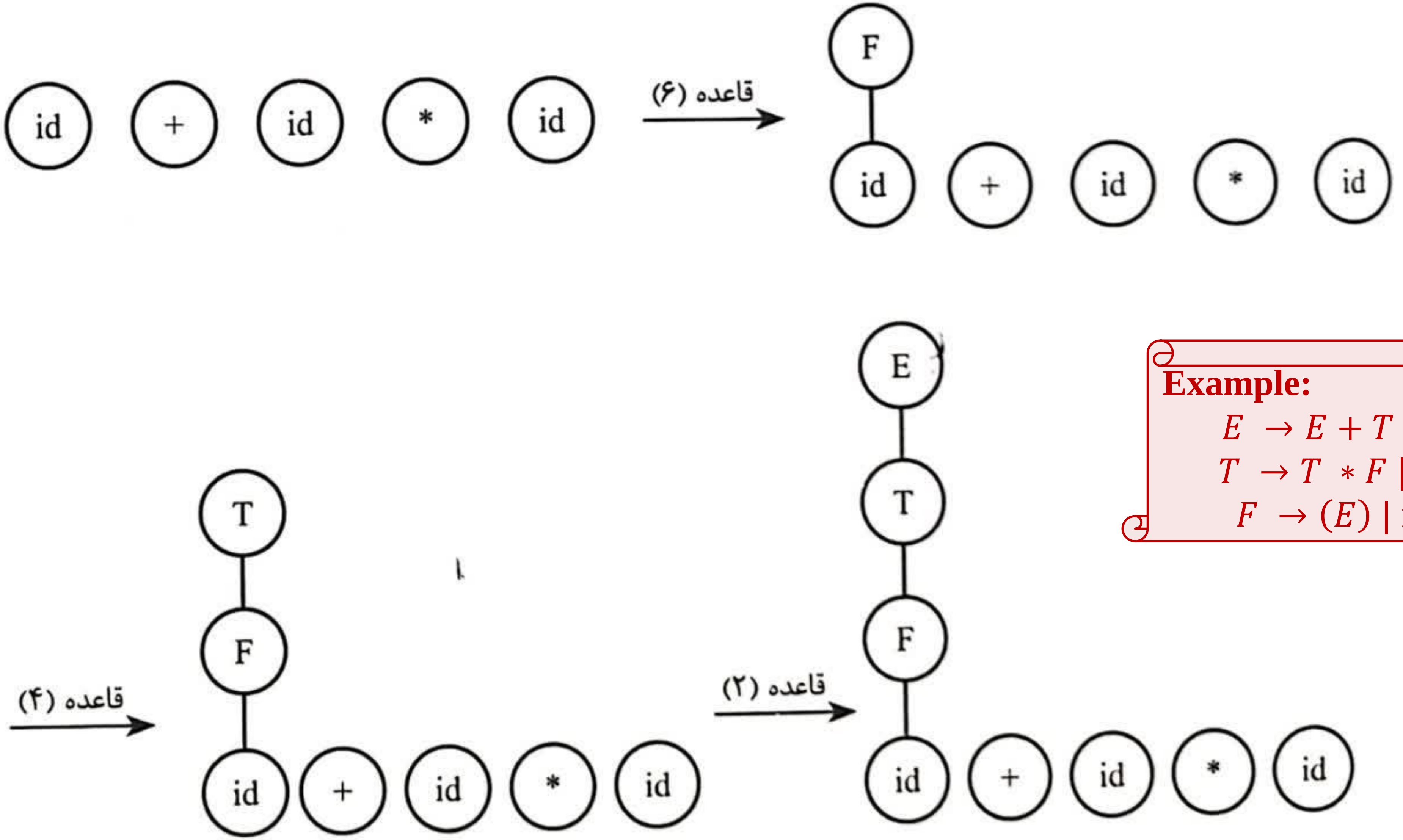


Example:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$

Bottom-Up Parsing...

عکس سمت راست‌ترین اشتقاق

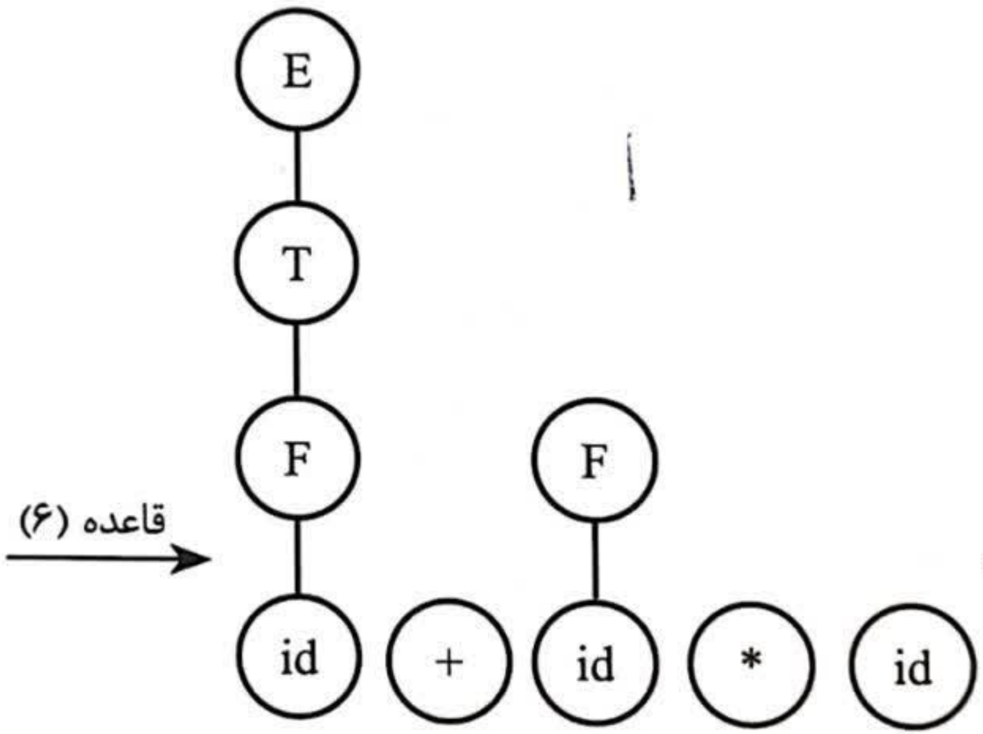


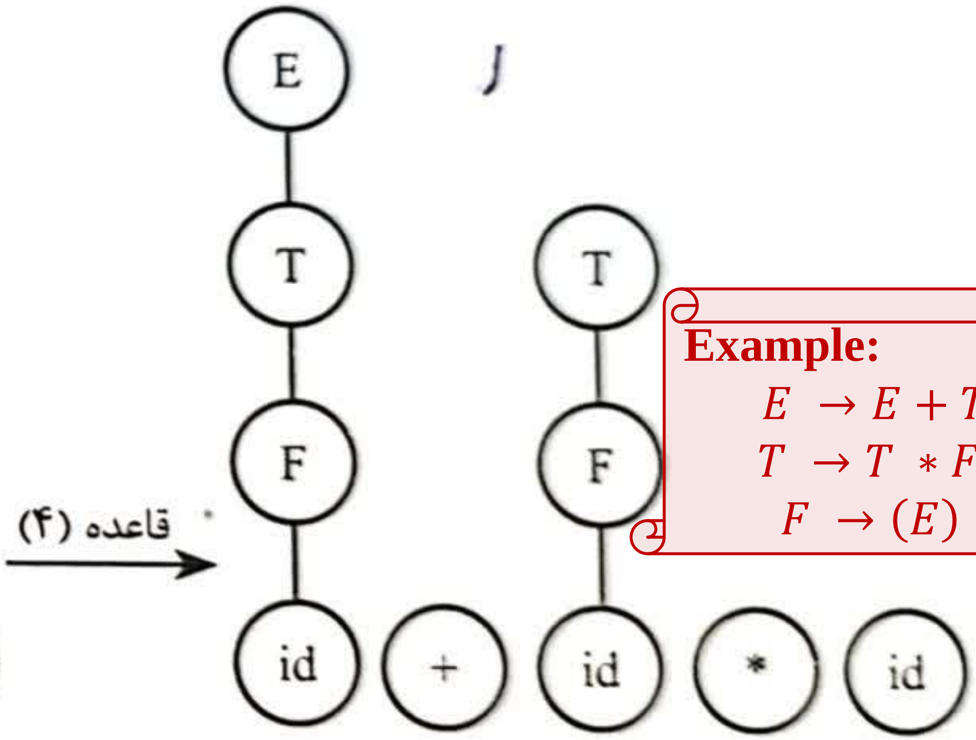
Example:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \mathrm{id}$$

Bottom-Up Parsing...

عکس سمت راست‌ترین اشتقاق



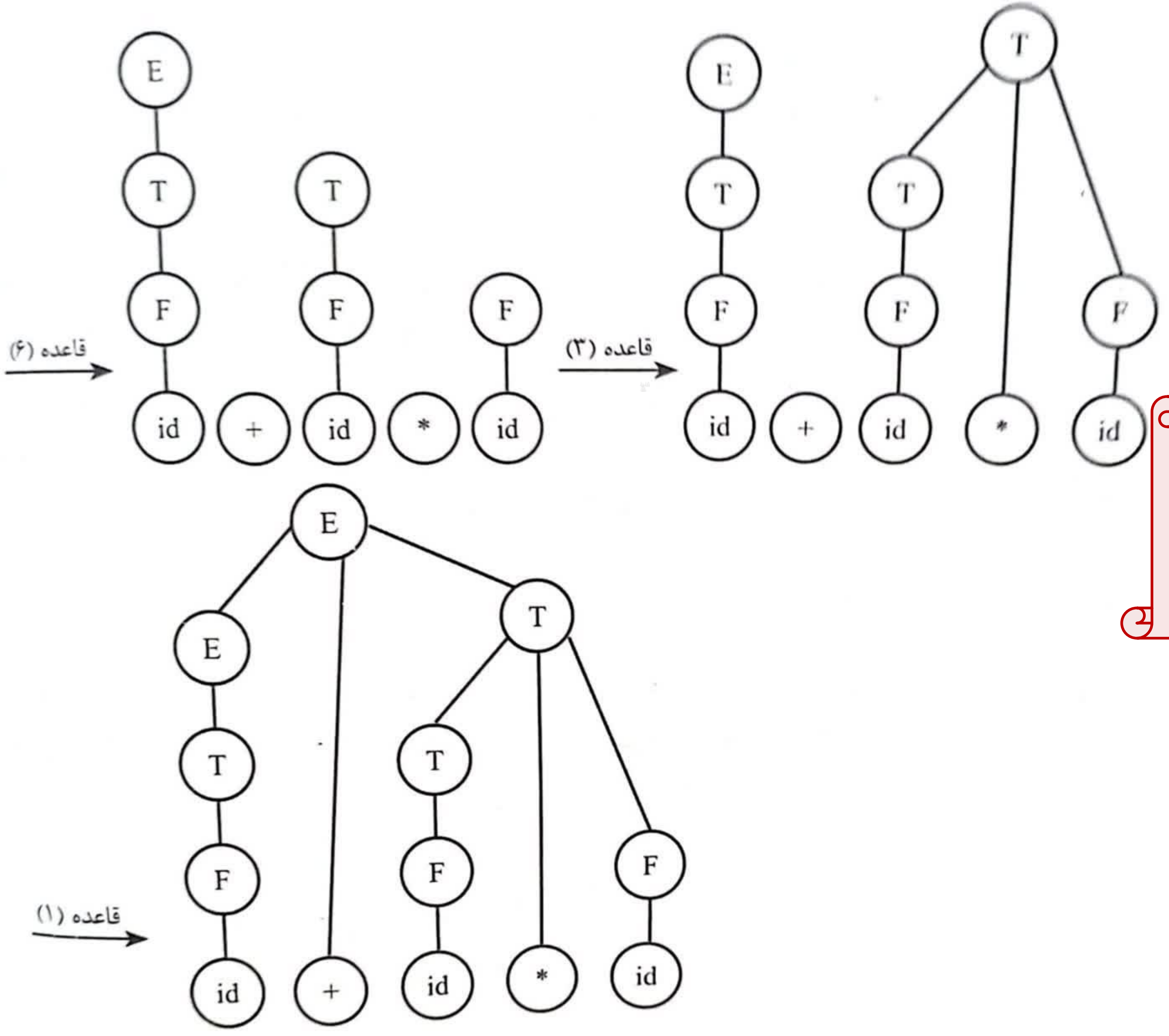


Example:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$

Bottom-Up Parsing...

عکس سمت راست‌ترین اشتقاق



Example:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$