CS 420 - Compilers

Dr. Chen-Yeou (Charles) Yu

- ~~Syntax Analysis (Ch 4)~~
  - ~~Introduction (Ch 4.1)~~
    - ~~The Role of the Parser (4.1.1)~~
    - ~~Representative Grammars (4.1.2)~~
    - ~~Syntax Error Handling (4.1.3)~~
    - ~~Error-Recovery Strategies (4.1.4)~~
  - Context-Free Grammars (4.2)
    - ~~The Formal Definition of a Context-Free Grammar (4.2.1)~~
    - ~~Notational Conventions (4.2.2)~~
    - Derivations (4.2.3)
    - Parse Trees and Derivations (4.2.4)
    - Ambiguity (4.2.5)
    - Verifying the Language Generated by a Grammar (4.2.6) (TBD, in Part4)

Derivations (4.2.3)

- In this section, we are reviewing some conventions in derivations.
- It could be kind of detail from what we previously learned but basically, they are the same.

Derivations (4.2.3)

- Derivations, productions are treated as rewriting rules.
- Beginning with the start symbol, each **rewriting** step **replaces** a **nonterminal** by the body of one of its productions.
- In the following example, we have a very special item
  - E derives -E $E \Rightarrow -E$

  - Both E or –E denote an expression
  - It is said “a replacement” of a single E by –E
  - The symbol “→” means, derives in one step
  - Compared to this $\overset{*}{\Rightarrow}$ symbol, it said derives in **zero or more steps**

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

Derivations (4.2.3)

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

- Let's prove the -(id) is a **particular instance** of an expression
  - E→-E→-(E)→-(id)
- Transitive rule:

1. $\alpha \overset{*}{\Rightarrow} \alpha$, for any string α, and
2. If $\alpha \overset{*}{\Rightarrow} \beta$ and $\beta \Rightarrow \gamma$, then $\alpha \overset{*}{\Rightarrow} \gamma$.

- Similarly, $\overset{+}{\Rightarrow}$ means, "derives in one or more steps."

Derivations (4.2.3)

- There are two ways of derivations
  - leftmost derivations
    - If a→b is a step in which the leftmost non-terminal in “a” is replaced, we write a → b with a “**lm**” under the “→”
  - rightmost derivations
    - Similarly, for the rightmost derivations, the rightmost nonterminal is always chosen. We write a → b with a “**rm**” under the “→”

Parse Trees and Derivations (4.2.4)

- A parse tree is a graphical representation of a derivation
- Each interior node of a parse tree represents the application of a production
- The interior node is labeled with the nonterminal A in the head of the production;
- The children of the node A are labeled, from left to right, by the symbols in the body of the production by which this A was replaced during the derivation.

Parse Trees and Derivations (4.2.4)

- There is an example here: (an example extended from 4.2.3)
- Based on this grammar:

$$E \rightarrow E + E \mid E * E \mid - E \mid (E) \mid \mathbf{id} \qquad (4.7)$$

- We know the –(id+id) can be derived in 2 ways:

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\mathbf{id}+E) \Rightarrow -(\mathbf{id}+\mathbf{id}) \qquad (4.8)$$

- Or,

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(E+\mathbf{id}) \Rightarrow -(\mathbf{id}+\mathbf{id}) \qquad (4.9)$$

- And because it is an “lm”, derivation, we can rewrite that as:

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

Parse Trees and Derivations (4.2.4)

- Here is the construction of our parsing tree, combining the formula 4.8 and 4.9
- Left to right
- This is the result of the parsing tree.
- In the next page, I will show you

the **process** of building this tree

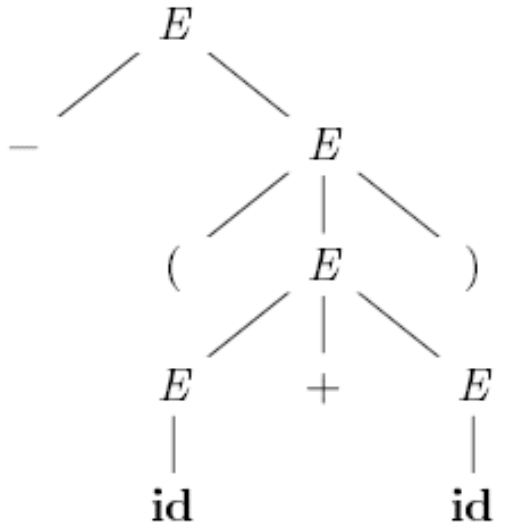

Figure 4.3: Parse tree for $-(\mathbf{id} + \mathbf{id})$

Parse Trees and Derivations (4.2.4)

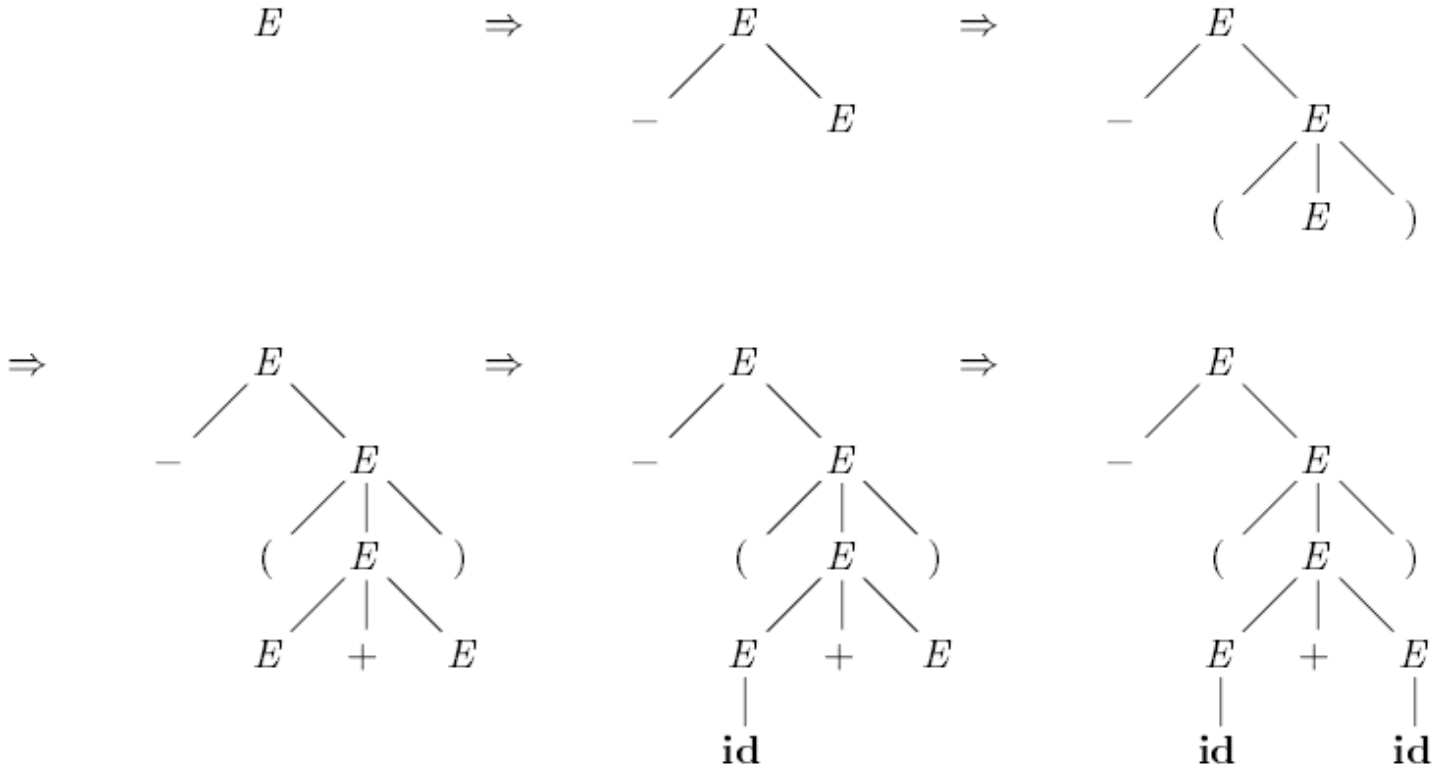

Figure 4.4: Sequence of parse trees for derivation (4.8)

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\mathbf{id}+E) \Rightarrow -(\mathbf{id}+\mathbf{id}) \qquad (4.8)$$

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(E+\mathbf{id}) \Rightarrow -(\mathbf{id}+\mathbf{id}) \qquad (4.9)$$

Ambiguity (4.2.5)

- **[Ambiguity]** The **definition** of ambiguous grammar is:
  - It can yield the **same final string**
  - It can construct 2 or more "**different parse trees**"

  Both **of the 2 conditions** are holding!
- The arithmetic expression grammar (4.3) permits two distinct leftmost derivations for the sentence id + id * id

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id} \qquad (4.3)$$

Ambiguity (4.2.5)

- It can yield 2 parsing trees
- One is thinking of the arithmetic

precedence and another is **not**

- In (b), it totally looks like

"+" is processed first, then,

Followed by "*"

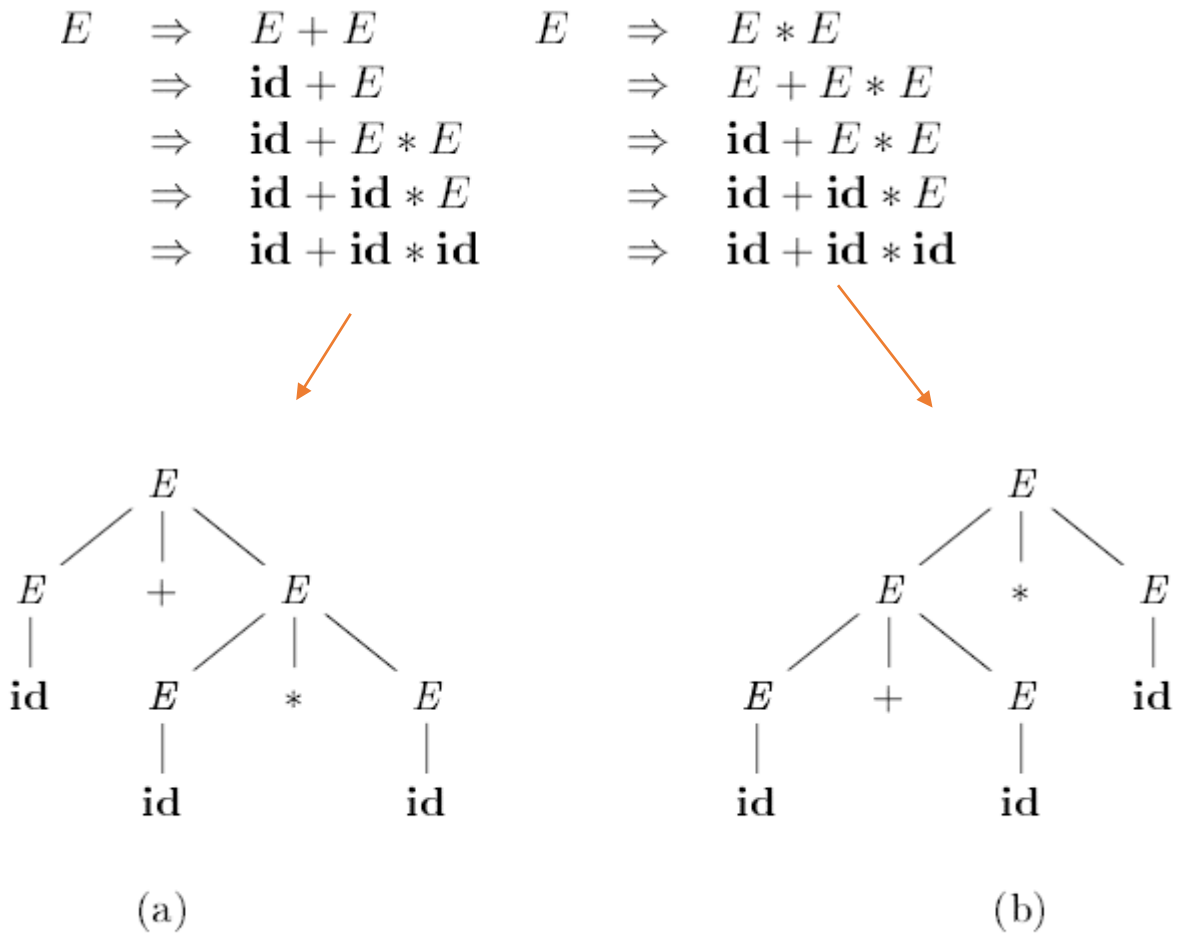

Figure 4.5: Two parse trees for **id+id*id**