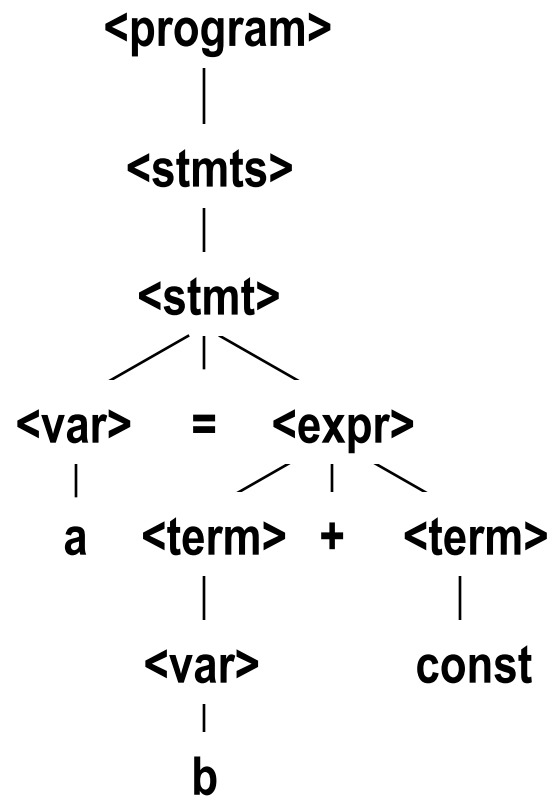
# Tutorial-3: part-I

Ambiguity

# Parse Tree

- A hierarchical representation of a derivation



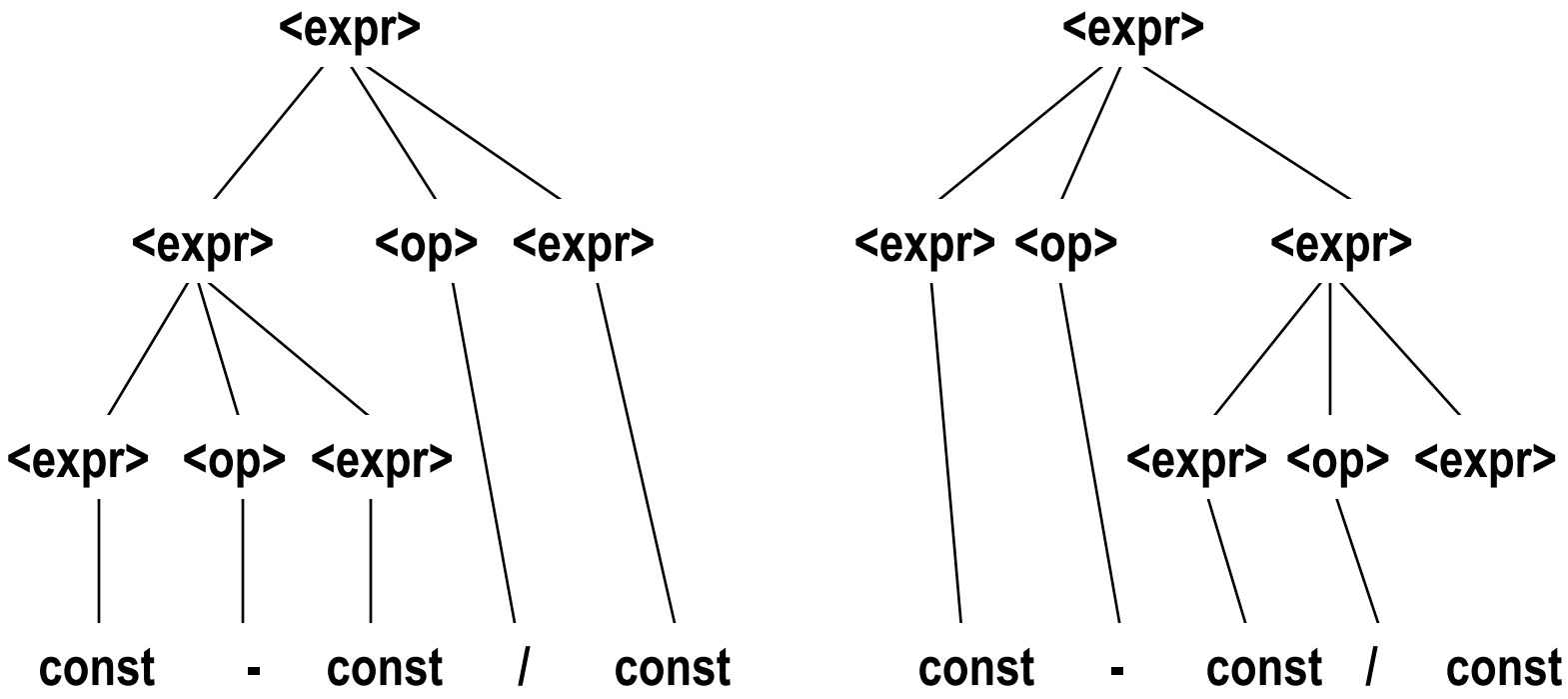
# Ambiguity in Grammars

- A grammar is *ambiguous* if and only if it generates a sentential form that has two or more distinct parse trees

# An Ambiguous Expression Grammar

<expr> → <expr> <op> <expr> | const

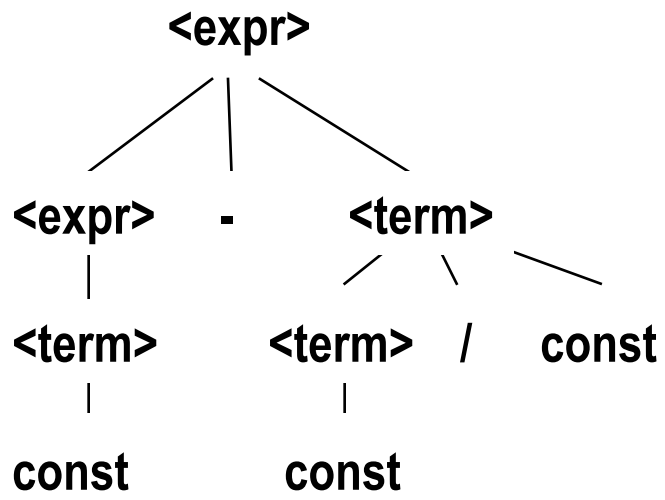
<op> → / | -



# An Unambiguous Expression Grammar

- If we use the parse tree to indicate precedence levels of the operators, we cannot have ambiguity (i.e., it must be eliminated)

<expr> → <expr> - <term> | <term>
<term> → <term> / const| const

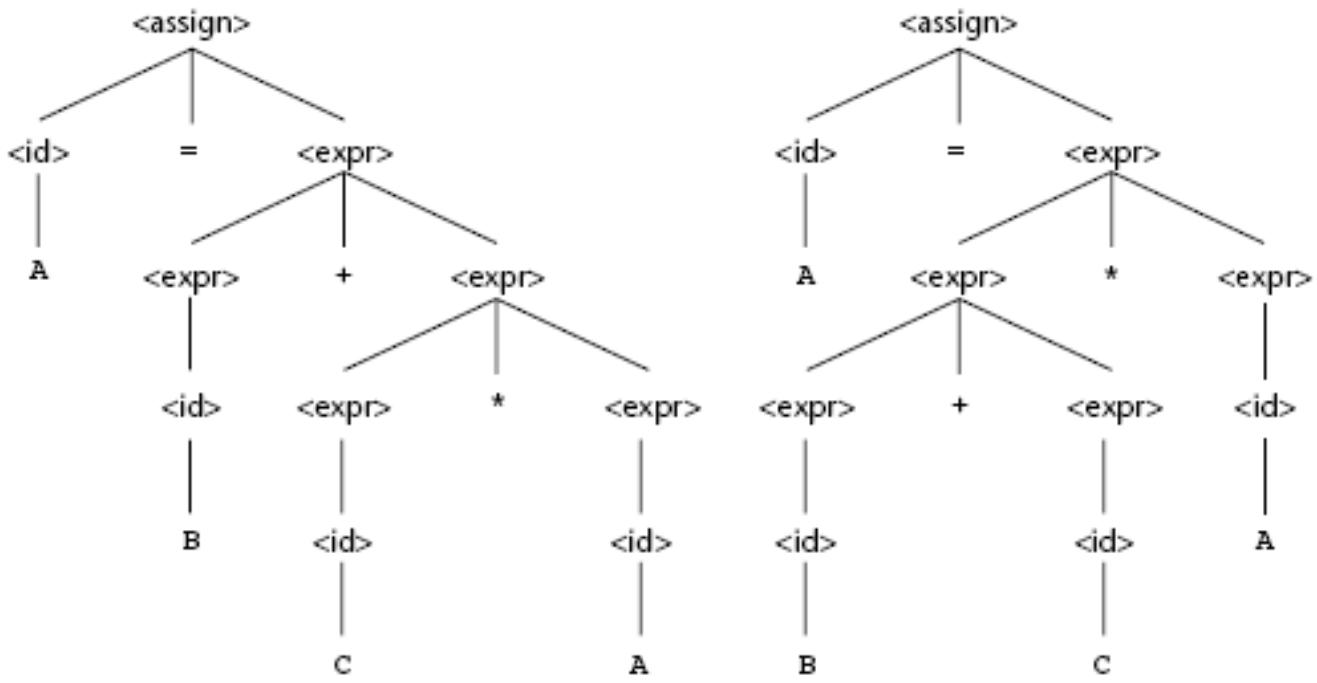


**EXAMPLE 3.3** An Ambiguous Grammar for Simple Assignment Statements

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <expr> + <expr>
         | <expr> * <expr>
         | ( <expr> )
         | <id>
```

**Figure 3.2**

Two distinct parse trees for the same sentence,
`A = B + C * A`



- Notice that the previous grammar allow the parse tree of an expression to grow on both the left and right sides.
- Ambiguity is a problem for compilers because they often base the semantics of these structures (e.g. the precedence of operators) on their syntactic structure.

# An un-ambiguous grammar that describes the same language

**EXAMPLE 3.2** — A Grammar for Simple Assignment Statements

<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <id> + <expr>
| <id> * <expr>
| ( <expr> )
| <id>

# Precedence

- Notice that the above grammar will always put the rightmost operation on the lowest level.
- So in the sentence A=A*B+C
- + will be in the lowest level and will have higher precedence than *

# Associtativity

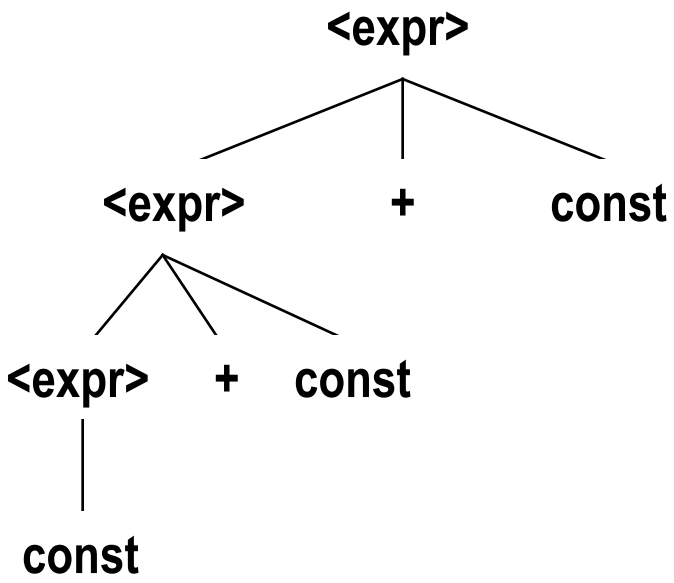
- Also in the sentence
- A=B+C+D
- The rightmost + will be at the lowest level
  ➔ incorrect associtativity

# Associativity of Operators

- Operator associativity can also be indicated by a grammar

```
<expr> -> <const>+<expr>|const   (right associatativity)

<expr> -> <expr> + const  |  const  (left associativity)
```



# Unambiguous and Correct Associativity (Left to right)

**EXAMPLE 3.4** An Unambiguous Grammar for Expressions

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <expr> + <term>
        | <term>
<term> → <term> * <factor>
        | <factor>
<factor> → ( <expr> )
          | <id>
```

**EXAMPLE 3.3** An Ambiguous Grammar for Simple Assignment Statements

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <expr> + <expr>
        | <expr> * <expr>
        | ( <expr> )
        | <id>
```