Syntax

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

Context-Free Grammar (CFG)

A CFG consists of

- A set of *terminals* T (basic alphabet)
- A set of *non-terminals* N
- A *start symbol* S (a non-terminal)
- A set of *production rules*

$$A ::= \omega$$

nonterminal

string of terminal and nonterminal

Derivation

A sequence from start symbol, each step a non-terminal is replace by RHS of production

A derivation of “xyz” with rule

```
var   ::= letter var
        | letter
```

var ⇒ letter var
⇒ letter letter var ⇒ letter letter letter
⇒ x letter letter ⇒ x y letter ⇒ x y z

A multi-step reduction $\quad$ var $\Rightarrow^*$ xyz

Derivation

$$\begin{aligned}\text{expr} &::= \text{id} \mid \text{number} \mid - \text{ expr} \\ &\quad \mid (\text{expr}) \mid \text{expr op expr} \\ \text{op} &::= + \mid - \mid * \mid / \\ &\dots\end{aligned}$$

A derivation of “x+3*y” with rule

$$\begin{aligned}\text{expr} &\Rightarrow \text{expr op expr} \Rightarrow \text{id op expr} \Rightarrow \text{id op expr op expr} \\ &\Rightarrow \text{id op number op expr} \Rightarrow \text{id op number op id} \\ &\Rightarrow \text{x op number op id} \Rightarrow \text{x} + \text{number op id} \\ &\Rightarrow \cdots \Rightarrow \text{x} + 3 * \text{y}\end{aligned}$$

A reduction $\text{expr} \Rightarrow^* \text{x} + 3 * \text{y}$

Order of Derivation

Derivation can follow any order

Leftmost derivation of “xyz”

$\text{var} \Rightarrow \text{letter var}$
$\quad \Rightarrow \text{x var} \Rightarrow \text{x letter var}$
$\quad \Rightarrow \text{x y var} \Rightarrow \text{x y letter} \Rightarrow \text{x y z}$

Rightmost derivation of “xyz”

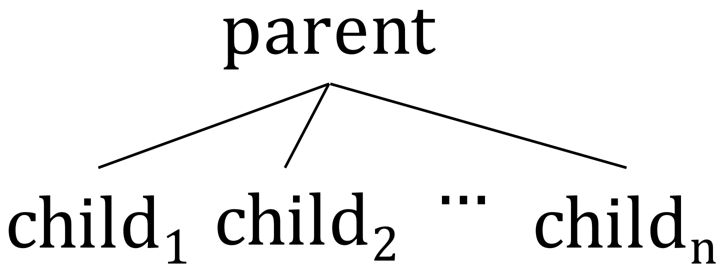
$\text{var} \Rightarrow \text{letter var}$
$\quad \Rightarrow \text{letter letter var} \Rightarrow \text{letter letter letter}$
$\quad \Rightarrow \text{letter letter z} \Rightarrow \text{letter y z} \Rightarrow \text{x y z}$

Parse Tree

Derivation in graphical from

Root: the start symbol

Leaf: terminal



represents

$$\mathrm{parent} \Rightarrow \mathrm{child}_1\ \mathrm{child}_2\ \ldots \mathrm{child}_n$$

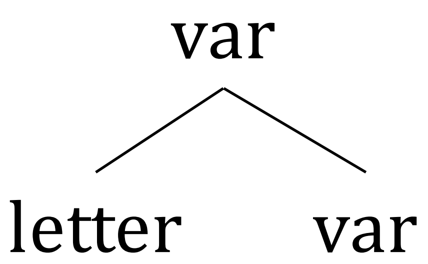
Parse Tree

Derivation in graphical from

Derivation Step

var ⇒ letter var

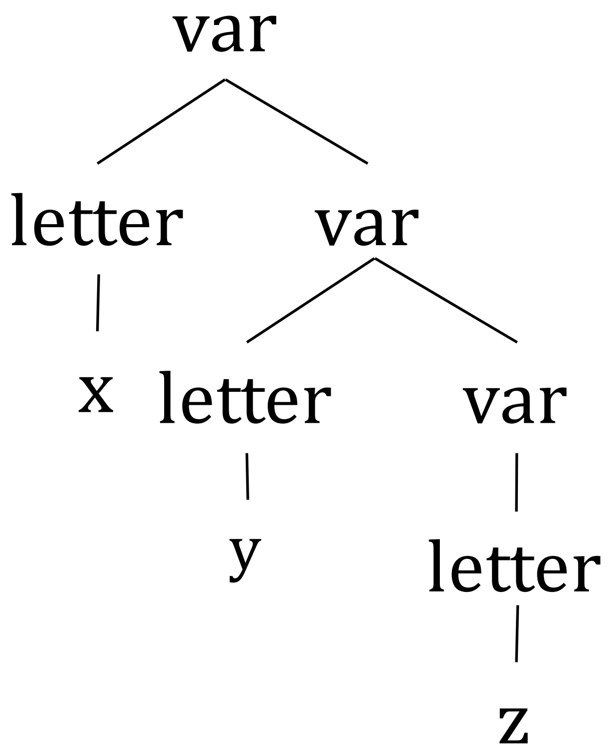
Tree



Parse Tree Example

A derivation of “xyz”

var ⇒ letter var
⇒ letter letter var | letter letter letter
⇒ x letter letter ⇒ x y letter ⇒ x y z



Parse tree for leftmost derivation?

Parse Tree Example

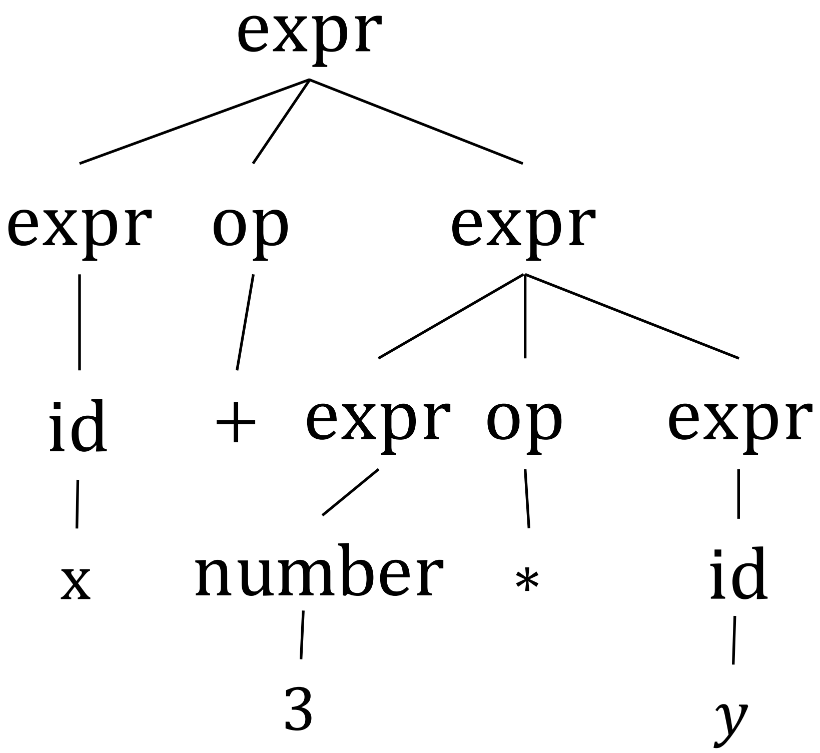
A derivation of "x+3*y"

$\text{expr} \Rightarrow \text{expr op expr} \Rightarrow \text{id op expr} \Rightarrow \text{id op expr op expr}$

$\quad \Rightarrow \text{id op number op expr} \Rightarrow \text{id op number op id}$

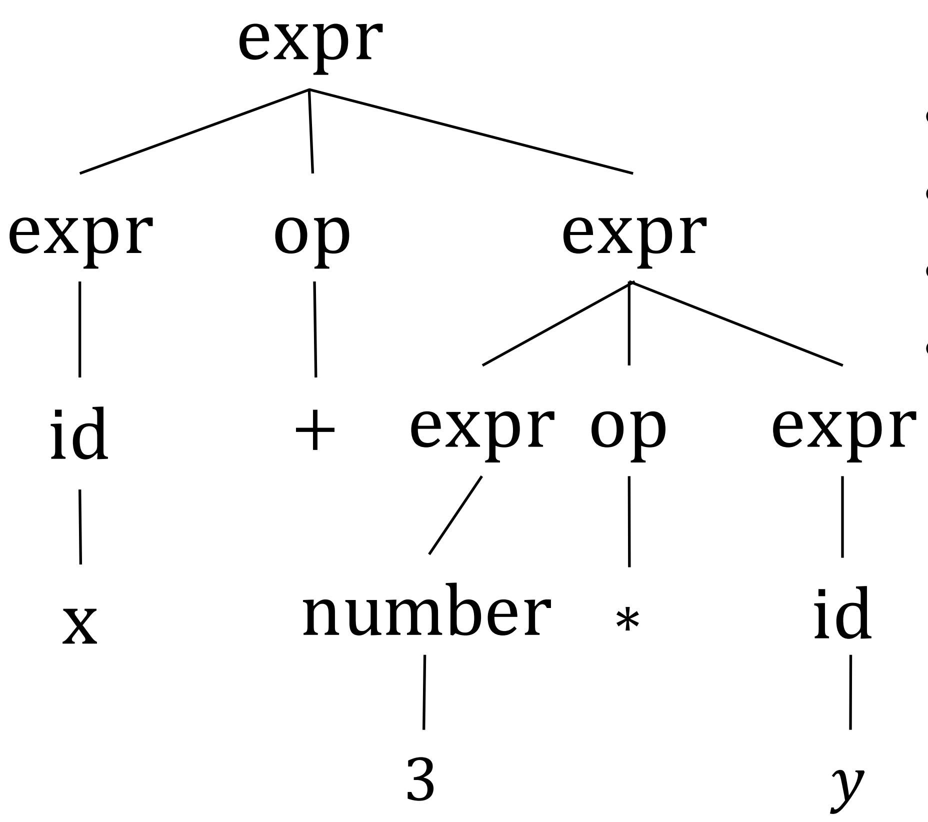
$\quad \Rightarrow \text{x op number op id} \Rightarrow \text{x} + \text{number op id}$

$\quad \Rightarrow \cdots \Rightarrow \text{x} + 3 \ * \text{y}$



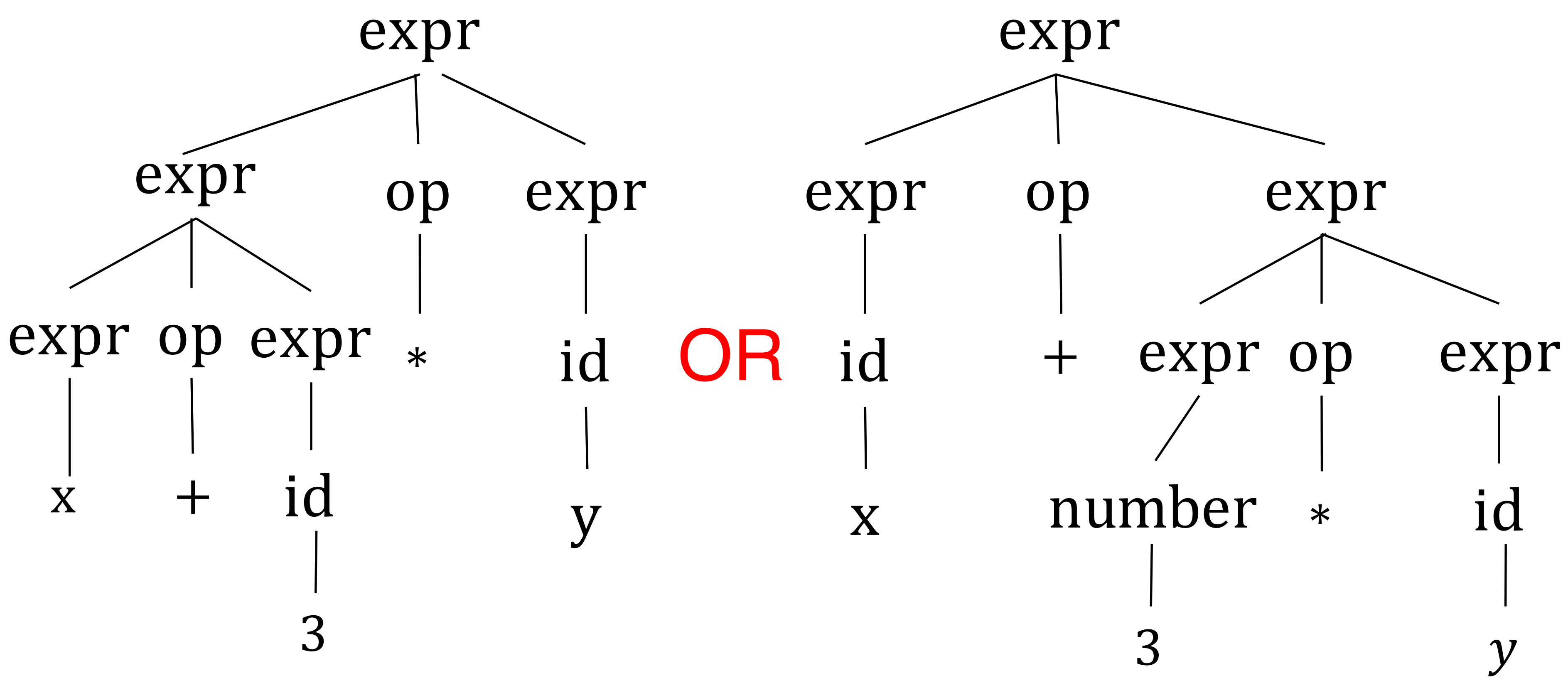
Parse tree for rightmost derivation?

Parse Tree Features



- Root is the start symbol
- Leaves are terminals
- Other nodes are nonterminals
- Leaves (left to right) form the parsed string

Ambiguity



A grammar is **ambiguous** if its language contains strings with two or more parse trees

Ambiguity

Is the following language ambiguous?

$$
\begin{aligned}
\text{expr} &\rightarrow \text{id} \mid \text{number} \mid - \ \text{expr} \\
&\quad \mid (\text{expr}) \mid \text{expr op expr} \\
\text{op} &\rightarrow + \mid -
\end{aligned}
$$

Precedence and Associativity

Precedence: which operator should be evaluated sooner

Associativity: operator with same precedence evaluated left-to-right or right-to-left

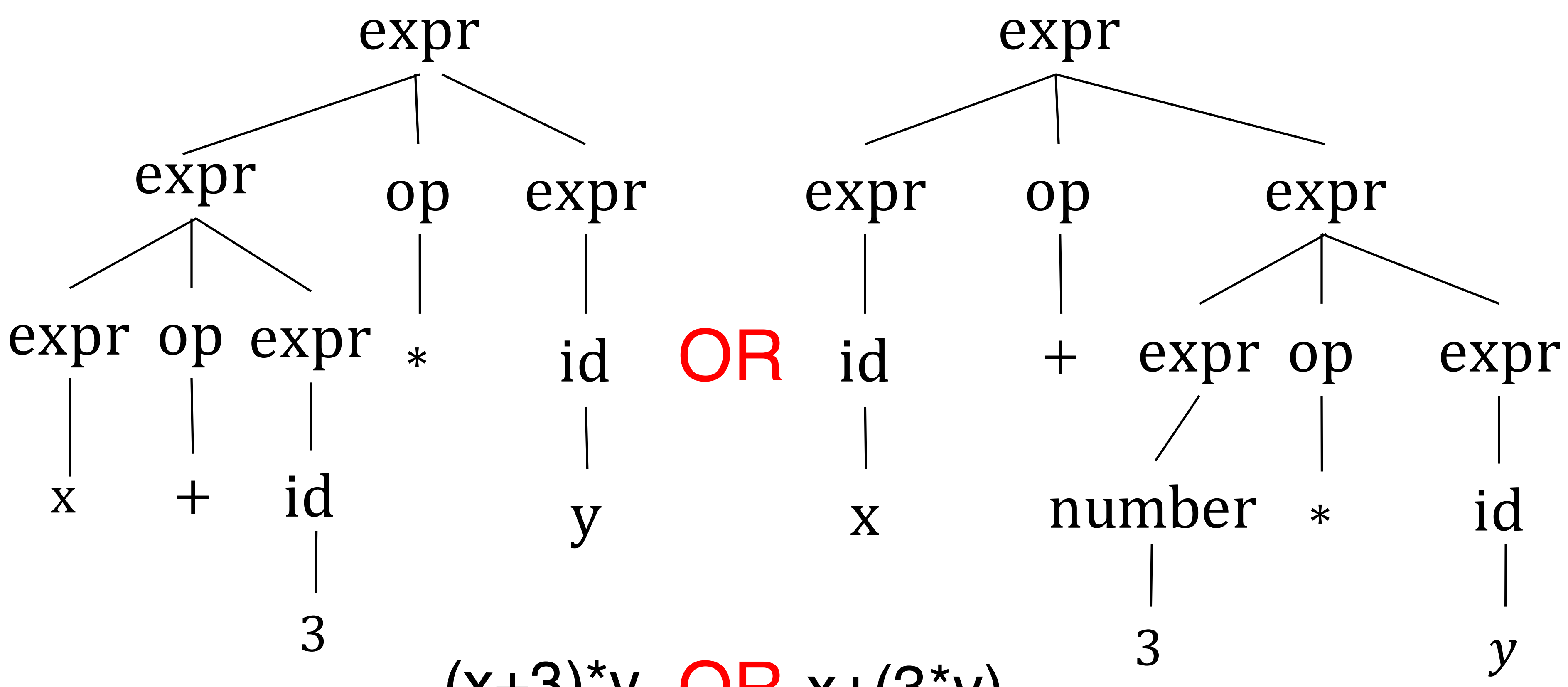
Arithmetic: * and / have higher precedence;
all operators are left-associative

Precedence and Associativity

$$
\begin{aligned}
\text{expr} &\rightarrow \text{id} \mid \text{number} \mid - \ \text{expr} \\
&\quad \mid (\text{expr}) \mid \text{expr op expr} \\
\text{op} &\rightarrow + \mid - \mid * \mid /
\end{aligned}
$$

The ambiguous grammar doesn't specify precedence and associativity

Precedence



(x+3)*y OR x+(3*y)

The lower an operation is, the higher precedence it has

Defining Precedence in Grammar

Define operations at different "levels"

$$\begin{aligned}\mathrm{expr} &\rightarrow \mathrm{id} \mid \mathrm{number} \mid -\,\mathrm{expr} \\ &\quad\mid (\mathrm{expr}) \mid \mathrm{expr}\ \mathrm{op}\ \mathrm{expr} \\ \mathrm{op} &\rightarrow + \mid - \mid * \mid /\end{aligned}$$

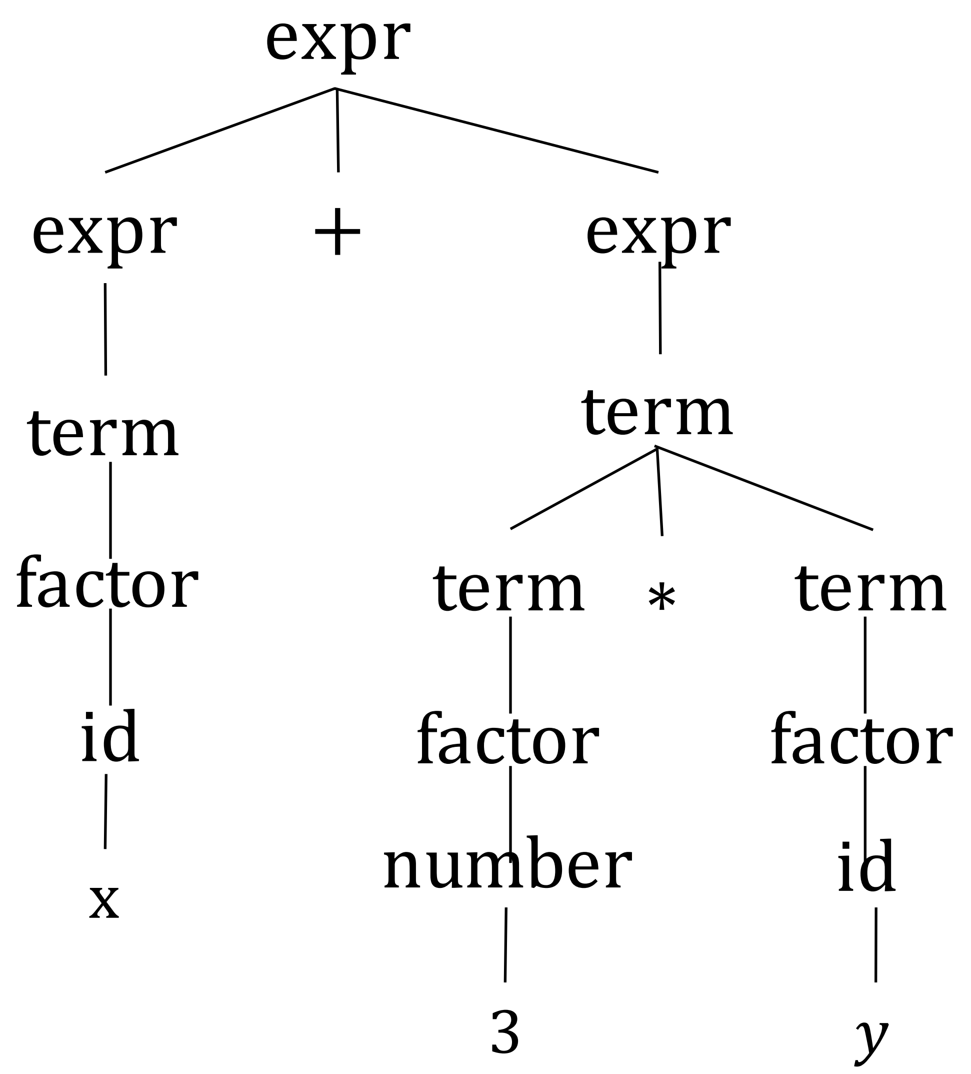
$$\Downarrow$$

$$\mathrm{expr} \rightarrow \mathrm{expr} + \mathrm{expr} \mid \mathrm{expr} - \mathrm{expr} \mid \mathrm{term}$$ Level1

$$\mathrm{term} \rightarrow \mathrm{term} * \mathrm{term} \mid \mathrm{term} / \mathrm{term} \mid \mathrm{factor}$$ Level2

$$\mathrm{factor} \rightarrow \mathrm{id} \mid \mathrm{number} \mid (\mathrm{expr}) \mid -\,\mathrm{factor}$$ Level3

The farther from start symbol, the higher precedence



x+(3*y)

Only one parse tree for x+3*y