Strings and Languages

Vocabulary(V): a finite set of symbols, e.g., $V = \{a, b, +, -\}$.

String (w): a finite sequence of symbols from V, e.g., $w_1 = a + a + a$ and $w_2 = + + b - -$

Empty String (λ): the string of length 0.

Concatenation (vw): vw is string v followed by string w.

w^n: string w repeated n times, e.g., $+^3 = +++$

V^*: all possible strings using V.

V^+: all possible strings using V except λ.

Language (L): A subset of V^*, e.g., $L =$ arithmetic expressions using V above

Phrase-Structure Grammars

A phrase-structure grammar G consists of:

- vocabulary V, divided into two subsets
  - *terminal symbols* T
  - *nonterminal symbols* N,
- *start symbol* $S \in N$
- a finite set of *productions* P

Each production has the form $x \rightarrow y$, where $x \in V^+$ and $y \in V^*$. This means that string uxv can be replaced with uyv.

---

w *derives* z ($w \overset{*}{\Rightarrow} z$) if

- $w = z$, or
- $w \Rightarrow z$, i.e., $w = uxv$ and $z = uyv$ and there is a production $x \rightarrow y$, or
- $w \Rightarrow w_2 \Rightarrow \ldots \Rightarrow z$ in a finite sequence.

The *language* of a grammar is:

$$L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}$$

Context-Free Grammars

A *context-free grammar* has productions:

$$A \to x$$

where A is nonterminal and $x \in V^*$

Example 1: Balanced Parentheses

$$S \to SS$$
$$S \to (S)$$
$$S \to \lambda$$

Example 2: Arithmetic Expressions (start E)

$$E \to I$$
$$E \to EOE$$
$$E \to (E)$$
$$I \to a \mid b \mid c$$
$$O \to + \mid - \mid * \mid /$$

Derivations and Trees

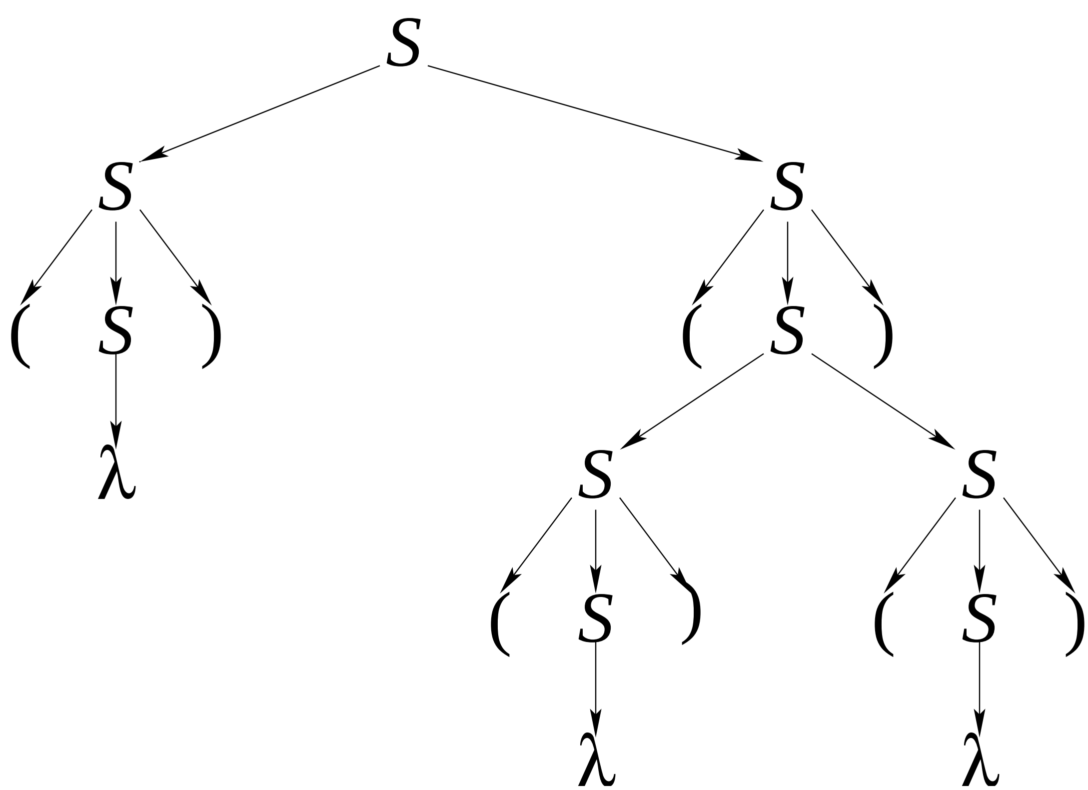
Leftmost Derivation of ()(()())

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()(SS)$$
$$\Rightarrow ()((S)S) \Rightarrow ()(()S) \Rightarrow ()(()(S)) \Rightarrow ()(()())$$

Rightmost Derivation of ()(()())

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S(SS) \Rightarrow S(S(S))$$
$$\Rightarrow S(S()) \Rightarrow S((S)()) \Rightarrow S(()()) \Rightarrow$$
$$(S)(()()) \Rightarrow ()(()())$$

Derivation Tree for ()(()())



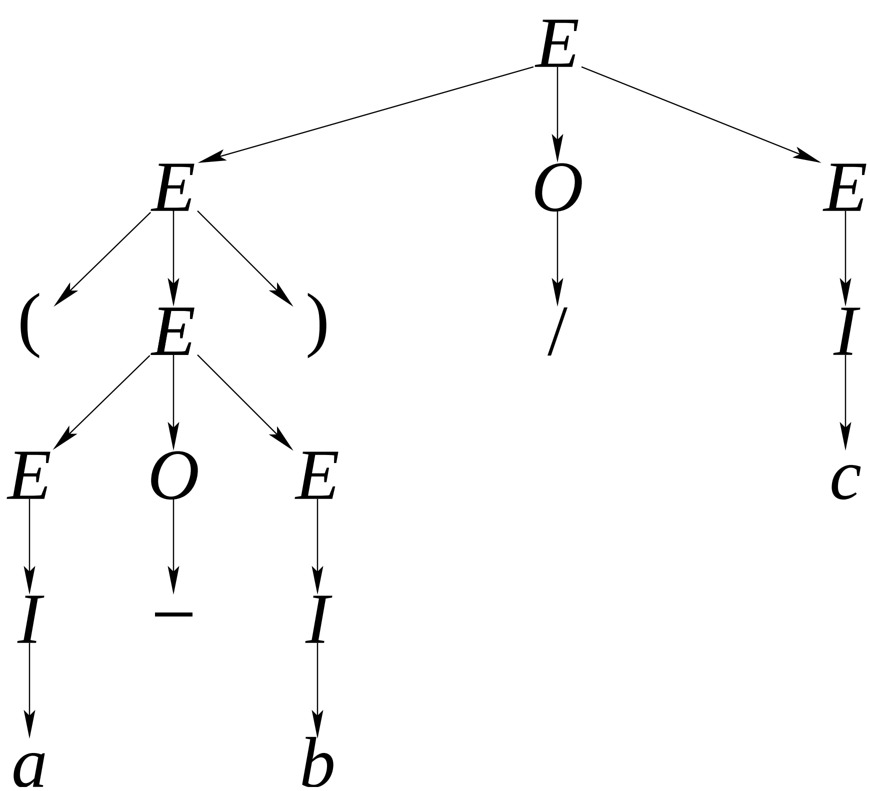
Leftmost Derivation of $(a-b)/c$

$$E \Rightarrow EOE \Rightarrow (E)OE \Rightarrow (EOE)OE \Rightarrow (IOE)OE \Rightarrow (aOE)OE \Rightarrow (a-E)OE \Rightarrow (a-I)OE \Rightarrow (a-b)OE \Rightarrow (a-b)/E \Rightarrow (a-b)/I \Rightarrow (a-b)/c$$

Rightmost Derivation of $(a-b)/c$

$$E \Rightarrow EOE \Rightarrow EOI \Rightarrow EOc \Rightarrow E/c \Rightarrow (E)/c \Rightarrow (EOE)/c \Rightarrow (EOI)/c \Rightarrow (EOb)/c \Rightarrow (E-b)/c \Rightarrow (I-b)/c \Rightarrow (a-b)/c$$

Derivation Tree for $(a-b)/c$



BNF (Backus-Naur Form) Grammars

BNF is a different notation for CFGs, traditionally used for programming languages.

Example 1: Balanced Parentheses

```
<parens> ::= <parens> <parens>
<parens> ::= ( <parens> )
<parens> ::= ""
```

Example 2: Arithmetic Expressions

```
<exp> ::= <id>
<exp> ::= <exp> <op> <exp>
<exp> ::= ( <exp> )
 <id> ::= a | b | c
 <op> ::= + | - | * | /
```