CMPS 3500

Programming Languages

Dr. Chengwei Lei
CEECS
California State University, Bakersfield
Extended BNF

- Optional parts are placed in brackets [ ]
  \[
  \langle \text{proc\_call} \rangle \rightarrow \text{ident} \ [ (\langle \text{expr\_list} \rangle ) ]
  \]

- Alternative parts of RHSs are placed inside parentheses and separated via vertical bars
  \[
  \langle \text{term} \rangle \rightarrow \langle \text{term} \rangle \ ( + | - ) \ \text{const}
  \]

- Repetitions (0 or more) are placed inside braces {}
  \[
  \langle \text{ident} \rangle \rightarrow \text{letter} \ \{ \text{letter|digit} \} 
  \]
BNF and EBNF

BNF

<expr> → <expr> + <term>
| <expr> - <term>
| <term>

EBNF
BNF and EBNF

- **BNF**
  \[
  <expr> \rightarrow <expr> + <term> \\
  \mid <expr> - <term> \\
  \mid <term>
  \]

- **EBNF**
  \[
  <expr> \rightarrow <term> \{ (+ | -) <term> \} 
  \]
BNF and EBNF

- **BNF**

  \[
  \langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle + \langle \text{term} \rangle \\
  \quad | \langle \text{expr} \rangle - \langle \text{term} \rangle \\
  \quad | \langle \text{term} \rangle \\
  \langle \text{term} \rangle \rightarrow \langle \text{term} \rangle * \langle \text{factor} \rangle \\
  \quad | \langle \text{term} \rangle / \langle \text{factor} \rangle \\
  \quad | \langle \text{factor} \rangle
  \]

- **EBNF**

  \[
  \langle \text{expr} \rangle \rightarrow \langle \text{term} \rangle \{(+ | -) \langle \text{term} \rangle\}
  \]
BNF and EBNF

- BNF

  \[ <\text{expr}> \rightarrow <\text{expr}> + <\text{term}> \]
  \[ \mid <\text{expr}> - <\text{term}> \]
  \[ \mid <\text{term}> \]

  \[ <\text{term}> \rightarrow <\text{term}> * <\text{factor}> \]
  \[ \mid <\text{term}> / <\text{factor}> \]
  \[ \mid <\text{factor}> \]

- EBNF

  \[ <\text{expr}> \rightarrow <\text{term}> \{ (+ \mid -) <\text{term}> \} \]
  \[ <\text{term}> \rightarrow <\text{factor}> \{ (* \mid /) <\text{factor}> \} \]
Recent Variations in EBNF

- Alternative RHSs are put on separate lines
- Use of a colon instead of =>
- Use of $\text{opt}$ for optional parts
- Use of $\text{oneof}$ for choices
Static Semantics

- Nothing to do with meaning
- Context-free grammars (CFGs) cannot describe all of the syntax of programming languages
- Categories of constructs that are trouble:
  - Context-free, but cumbersome (e.g., types of operands in expressions)
  - Non-context-free (e.g., variables must be declared before they are used)
A context-free grammar $G$

- A context-free grammar $G$ is defined by the 4-tuple:
  - $G = (V, \Sigma, R, S)$ where
    - $V$ is a finite set; each element $v \in V$ is called a nonterminal character or a variable. Each variable represents a different type of phrase or clause in the sentence. Variables are also sometimes called syntactic categories. Each variable defines a sub-language of the language defined by $G$.
    - $\Sigma$ is a finite set of terminals, disjoint from $V$, which make up the actual content of the sentence. The set of terminals is the alphabet of the language defined by the grammar $G$.
    - $R$ is a finite relation from $V$ to $(V \cup \Sigma)$, where the asterisk represents the Kleene star operation. The members of $R$ are called the (rewrite) rules or productions of the grammar.
    - $S$ is the start variable (or start symbol), used to represent the whole sentence (or program). It must be an element of $V$. 