

06 - Syntax Analysis

Dr. Robert Lowe

Division of Mathematics and Computer Science
Maryville College

Outline

- 1 Syntax Analysis
- 2 Example: L Programming Language

Syntax Analyzer

- There are two main parts to syntax analysis:
 - **Lexing** - Process the micro-syntax of the language.
 - **Syntax Analysis** - Process the context-free syntax of the language.
- The syntax analyzer can be created directly from the BNF specification of a language.

Lexical Analysis Abstraction

For now our lexer will consist of the following:

- A global variable `symbol`
- **procedure** `next_symbol`
 - 1 Place the next basic symbol in the global variable.
 - 2 Advance the input stream.
- **procedure** `mustbe (s)`
 - 1 if `s` is the `symbol`, call `next_symbol`
 - 2 Otherwise, report an error.
- **procedure** `have (s)`
 - 1 if `s` is the `symbol`, call `next_symbol` and return `true`.
 - 2 Otherwise, return `false`

Coding from BNF

- A production like this: $\langle \rangle ::= a\langle A \rangle$

- Would be coded:

```
mustbe("a"); A();
```

- A production like this: $\langle \rangle ::= a\langle A \rangle \mid b\langle B \rangle$

- Would be coded:

```
if have("a") then A() else { mustbe("b");  
B() }
```

- A production like this: $\langle \rangle ::= a\langle A \rangle \mid b\langle B \rangle \mid c \langle C \rangle$

- Would be coded as:

```
if( have("a") ) { A(); }  
else if( have("b") ) { B(); }  
else { mustbe("c"); C(); }
```

Repetition Productions

$\langle \rangle ::= \langle A \rangle [b\langle A \rangle]^*$ (Where * means repeat “zero or more times”)

```
do {  
    A();  
} while (have ("b"));
```

The Grammar of L

$\langle \text{program} \rangle ::= \langle \text{expression} \rangle$

$\langle \text{expression} \rangle ::= \langle \text{term} \rangle \langle \text{expression-tail} \rangle$

$\langle \text{expression-tail} \rangle ::= \lambda \mid '+' \langle \text{term} \rangle \langle \text{expression-tail} \rangle$

$\langle \text{term} \rangle ::= \langle \text{factor} \rangle \langle \text{term-tail} \rangle$

$\langle \text{term-tail} \rangle ::= \lambda \mid '*' \langle \text{factor} \rangle \langle \text{term-tail} \rangle$

$\langle \text{factor} \rangle ::= \langle \text{unit} \rangle \mid '(' \langle \text{expression} \rangle ')'$

$\langle \text{unit} \rangle ::= '0' \mid '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7' \mid '8' \mid '9'$

Activity: Let's create a syntax analyzer for L!