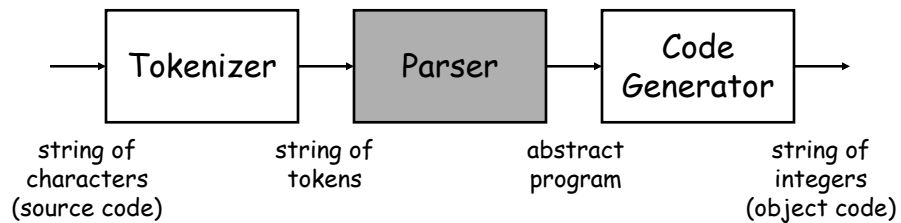


Translator Architecture



Here's the Plan

- Design a context-free grammar to specify (syntactically) valid BL programs
- Use the grammar to implement a recursive descent parser (i.e., an algorithm to parse BL programs and construct the corresponding Program object)

Specifying Syntax with CFGs

- Context-free grammars (cfg)
- Some syntactically valid real constants:
 - 37.044
 - 615.22E16
 - 99241.
 - 18.E-93

Rewrite Rules for Real Constants

$\langle \text{real constant} \rangle \rightarrow \langle \text{digit seqnce} \rangle . \langle \text{digit seqnce} \rangle \mid$
 $\langle \text{digit seqnce} \rangle . \langle \text{digit seqnce} \rangle \langle \text{exponent} \rangle \mid$
 $\langle \text{digit seqnce} \rangle . \mid$
 $\langle \text{digit seqnce} \rangle . \langle \text{exponent} \rangle$

$\langle \text{exponent} \rangle \rightarrow E \langle \text{digit seqnce} \rangle \mid$
 $E + \langle \text{digit seqnce} \rangle \mid$
 $E - \langle \text{digit seqnce} \rangle$

$\langle \text{digit seqnce} \rangle \rightarrow \langle \text{digit} \rangle \langle \text{digit seqnce} \rangle \mid$
 $\langle \text{digit} \rangle$

$\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Four Components of a CFG

- Nonterminal symbols
 - $\langle \text{real constant} \rangle$, $\langle \text{exponent} \rangle$,
 $\langle \text{digit seqnce} \rangle$, $\langle \text{digit} \rangle$
- Terminal symbols
 - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E, +, -, .
- Start symbol
 - $\langle \text{real constant} \rangle$
- Rewrite rules
 - on previous slide

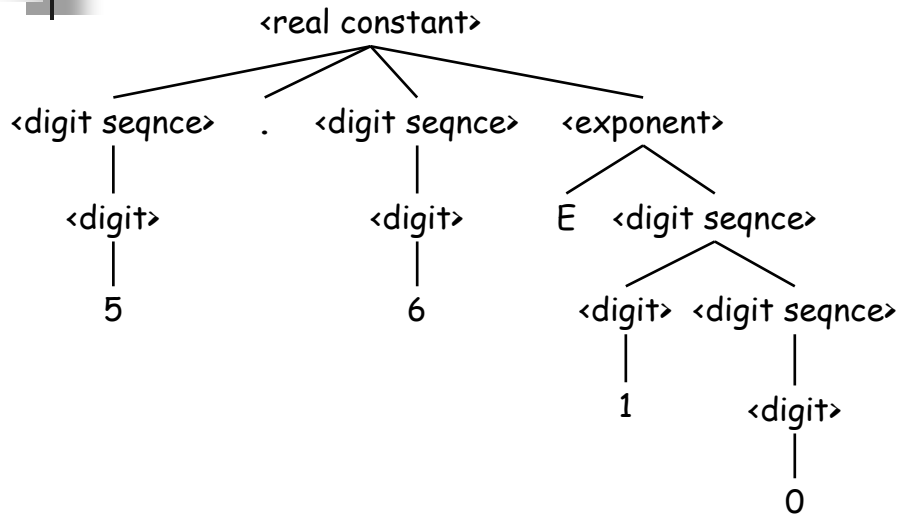
A Derivation of 5.6E10

$\langle \text{real constant} \rangle \Rightarrow \langle \underline{\text{digit seqnce}} \rangle . \langle \text{digit seqnce} \rangle \langle \text{exponent} \rangle$
 $\Rightarrow \langle \underline{\text{digit}} \rangle . \langle \text{digit seqnce} \rangle \langle \text{exponent} \rangle$
 $\Rightarrow 5 . \langle \underline{\text{digit seqnce}} \rangle \langle \text{exponent} \rangle$
 $\Rightarrow 5 . \langle \underline{\text{digit}} \rangle \langle \text{exponent} \rangle$
 $\Rightarrow 5 . 6 \langle \underline{\text{exponent}} \rangle$
 $\Rightarrow 5 . 6 E \langle \underline{\text{digit seqnce}} \rangle$
 $\Rightarrow 5 . 6 E \langle \underline{\text{digit}} \rangle \langle \text{digit seqnce} \rangle$
 $\Rightarrow 5 . 6 E 1 \langle \underline{\text{digit seqnce}} \rangle$
 $\Rightarrow 5 . 6 E 1 \langle \underline{\text{digit}} \rangle$
 $\Rightarrow 5 . 6 E 1 0$

If it's possible to find such a derivation, we write:

$\langle \text{real constant} \rangle \stackrel{*}{\Rightarrow} 5.6E10$

A Derivation Tree for 5.6E10



Find a Derivation Tree for 5.E3

- The derivation tree:

- Can you find a derivation tree for .6E10?

Language Generated by a CFG

- **Definition:** Let $G = (\text{Nonterminals, Terminals, } \langle \text{start symbol} \rangle, \text{Rewrite Rules})$ be a context-free grammar. The language *generated* (or *specified*) by G is denoted $L(G)$ and is defined as:

$$L(G) = \{x: \text{string of Terminals } (\langle \text{start symbol} \rangle \xRightarrow{*} x)\}$$

Another Example:

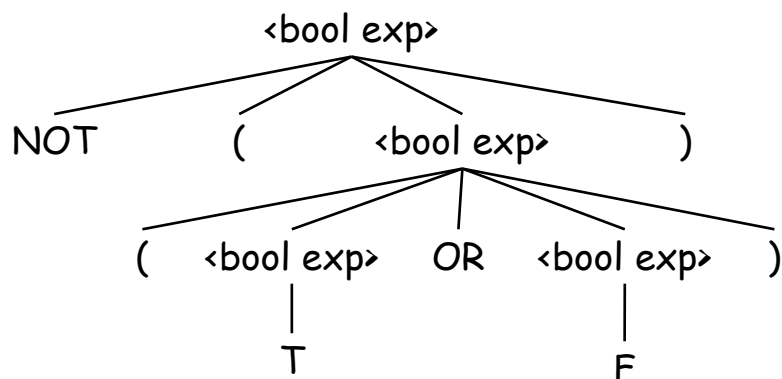
A CFG for Boolean Expressions

$\langle \text{bool exp} \rangle \rightarrow T \mid$
 $F \mid$
 $\text{NOT} (\langle \text{bool exp} \rangle) \mid$
 $(\langle \text{bool exp} \rangle \text{ AND } \langle \text{bool exp} \rangle) \mid$
 $(\langle \text{bool exp} \rangle \text{ OR } \langle \text{bool exp} \rangle)$

CFG for Boolean Expressions

- What are the nonterminal symbols?
- What are the terminal symbols?
- What is the start symbol?
- How many rewrite rules?

A Derivation Tree for NOT((T OR F))



*Find a Derivation Tree for
((T OR NOT (F)) AND T)*

A Famous Context-free Grammar

$\langle \text{expression} \rangle \rightarrow \langle \text{expression} \rangle \langle \text{addop} \rangle \langle \text{term} \rangle \mid \langle \text{term} \rangle$
 $\langle \text{term} \rangle \rightarrow \langle \text{term} \rangle \langle \text{multop} \rangle \langle \text{factor} \rangle \mid \langle \text{factor} \rangle$
 $\langle \text{factor} \rangle \rightarrow (\langle \text{expression} \rangle) \mid \langle \text{digit seqnce} \rangle$
 $\langle \text{addop} \rangle \rightarrow + \mid -$
 $\langle \text{multop} \rangle \rightarrow * \mid \text{DIV} \mid \text{MOD}$
 $\langle \text{digit seqnce} \rangle \rightarrow \langle \text{digit} \rangle \langle \text{digit seqnce} \rangle \mid \langle \text{digit} \rangle$
 $\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

What's So Special

About This CFG?

- Find a derivation tree for $4 + 6 * 2$

What's So Special Continued...

- Find the derivation tree for $(4 + 6) * 2$

How About These Rewrite Rules?

$\langle \text{expression} \rangle \rightarrow \langle \text{expression} \rangle \langle \text{op} \rangle \langle \text{expression} \rangle \mid$
 $(\langle \text{expression} \rangle) \mid$
 $\langle \text{digit seqnce} \rangle$

$\langle \text{op} \rangle \rightarrow + \mid - \mid * \mid \text{DIV} \mid \text{MOD}$

$\langle \text{digit seqnce} \rangle \rightarrow \langle \text{digit} \rangle \langle \text{digit seqnce} \rangle \mid$
 $\langle \text{digit} \rangle$

$\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

New Rules Continued...

- Find a derivation tree for $4 + 6 * 2$