Programming Languages: Syntax

Sudarshan S. Chawathe

2025-09-12

School of Computing and Information Science
& Climate Change Institute
University of Maine

Announcements and Reminders

- Sound and visuals check.
- Main online resource: Class Web site:
  - http://chaw.eip10.org/cos301/
  - also linked from my Web page, etc.
- Homework HW01 posted on class Web site.
  - Work early, work often.
- Syllabus:
  - Posted on (and is most of) main Web site.
  - Gen AI policy.
- Brightspace for some things only.
  - discussion forum: Please use!
  - homework and other submissions.

Plan for today

- Continuing where we left off.
- (Recall) Material mostly from beginning of Chapter 2 of the textbook.
  - (Review) What is *syntax* (for programming languages)?
  - (Review) What are some standard ways of specifying syntax?
  - *Regular expressions*, *[E]BNF*, *CFG*.
  - Theory of CFGs etc.
  - *Parse trees* and *abstract syntax trees (AST)*.
- (Review) Bigger picture question (related to homework):
  - How to implement a simple language like:
    - x = 5 + 3
    - y = 48 / (4 * 4)
    - z = x + 2 * y
    - etc.

(Review) The semantics of syntax and semantics

- syntax
  - appearance
  - superficial structure
  - examples
    - foo(42); v. (foo 42)
    - if foo then bar; else baz; v. (if foo bar baz)
  - can be *statically* checked
    - statically = without running the program, usually at compile time.
- semantics
  - meaning
  - deep structure
  - examples
    - (f0 (f1) (f2)) in Common Lisp v. Scheme.
  - *may not* be statically checkable
- more complex than above, but OK for now.

(Review) Specifying syntax: context-free grammars (CFG)

- A formal method for specifying syntax
  - not the only way, but most widely used.
  - because it has just about the right amount of *expressive power*
    - regular expressions: not enough (for typical PLs)
    - context-sensitive grammars, Turing machines, etc.: too much
- A PL's syntax is specified by a CFG
  - but how is the CFG specified?
  - and then how is that specified?
  - . . . ?
- A *metalanguage* specifies a language.
  - a meta meta language specifies a metalanguage
  - . . .
  - at some point it is simple enough that we can stop (we hope!)
    - "It's turtles all the way down!"

(Review) Context-free grammars (CFG)

- Contrast with context-sensitive grammars.
  - informally, those can say things like whether "blue" qualifies as a "color" depends on the context in which "color" is used.
  - very interesting but we won't pursue here.
- In a CFG whether "blue" is a "color" cannot depend on the context in which "color" is used.
- Specified using (E)BNF
  - Extended Backus-Naur Form
  - not the only way, but most common

(Review) Terminals and nonterminals

- *terminals* or *tokens*
  - $\approx$ granules of the program source that are not analyzed internally by the CFG
    - but may be analyzed internally by the *lexer*.
  - examples
    - $=$
    - ;
    - avonum
    - 6.022E23
- *nonterminals* or *syntactic categories*
  - have components that are specified and analyzed by a CFG
  - examples
    - *assignment statement*: $\text{avonum} = 6.023;$
    - *return statement*: $\text{return } 42;$
    - *predicate*: $\text{avonum} > 42$

(Review) BNF: Backus-Naur Form(at)

- BNF spec = *set* of rules
  - N.B.: above spec is in a meta meta language.
- Each rule has the form:
  - *nonterminal* ::= sequence of *terminals* and *nonterminals*
    - again a meta-meta-language spec.
    - *::=* = "is" or "is composed of" or "can be replaced by"
- Examples
  - *assignment-statement* ::= *variable-name assignment-operator rval ;* ;
    - meta-language (BNF): ::=, ;
    - language: (C-like): *;*
  - *assignment-operator* ::= = ;
  - *statements* ::= *statement statements* | ;
    - | is short-hand for multiple rules with same LHS.

(Review) BNF example

- from textbook

<primitive-type> ::= boolean | char | byte | short | int | long | float | ...
<argument-list> ::= <expression> | <argument-list> , <expression>
<selection-statement> ::= if (<expression>) <statement>
^^I^^I^^I| if (<expression>) <statement> else <statement>
^^I^^I^^I| switch (<expression>) <block>
^^I^^I^^I;

- Exercises
  - For each component above: Is it in language or meta-language?
  - Describe in English as precisely as possible.
  - Provide illustrative examples (in the *language*) making reasonable assumptions.

EBNF = Extended BNF

- BNF + some convenience features
- foo? or [foo] = optional foo
  - exercise: language v. metalanguage elements above
- foo* or {foo} = a sequence of zero or more foo
- foo+ = a sequence of one or more foo
- parentheses (in metalanguage; language may have them too!)
  - (foo bar)+ = sequence of one or more instances of foo bar
  - (*(* *)*)+ = sequence of one or more *()*.
    - !!

CFG formally

- $G = (N, T, P, S)$
  - N: a set of symbols (*nonterminals*)
  - T: another set of symbols (*terminals*)
    - $N \cap T = \emptyset$
  - P: set of *productions*
    - each of form $n \rightarrow \alpha$ where
    - where $n \in N$ and $\alpha \in \{N \cup T\}^*$
  - $S \in N$: special nonterminal called *start*

CFG for infix expressions

- example from the textbook