Programming Languages: Syntax

Sudarshan S. Chawathe

2025-09-08

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

Announcements and Reminders

- Sound and visuals check.
- Introductions (continued).
- Read the textbook:
  - Kent D. Lee. Foundations of Programming Languages. Undergraduate Topics in Computer Science. Springer Nature, 2nd edition, December10 2017
  - Sections as announced.
- Syllabus.
  - will be posted on (and is most of) main Web site:
  - http://chaw.eip10.org/cos301/
    - also linked from my Web page, etc.
- Brightspace for some things only.
  - discussion forum.
  - homework and other submissions.
  - not yet active.

Plan for today

- Scanning (*lexing*) and parsing (*yaccing*).
  - huh? what? why? how?
- Material mostly from beginning of Chapter 2 of the textbook.
  - What is *syntax* (for programming languages)?
  - What are some standard ways of specifying syntax?
  - *Regular expressions*, *[E]BNF*, *CFG*.
  - Theory of CFGs etc.
  - *Parse trees* and *abstract syntax trees (AST)*.
- 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.

Show me the code!

- Example: lexing and yaccing in python using *PLY*.
  - Just a peek/teaser today; more a bit later.
- Switch to code.

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.

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!"

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