6.110 Computer Language Engineering

Recitation 3: Parser Generators Demo

February 14, 2025

Announcements and updates ←

Parser generator demo

Upcoming deadlines

- Mini-quiz 2 and Weekly Check-in 2 have been released, due **Thursday, February 20**
- Project phase 1 is due **Friday, February 21**
  - No late submissions!
- Project phase 2 will be released on the same day, due two weeks after (Friday, March 7)
- Team submission form due **Wednesday, February 19**

Coming up soon... **Week 3**

Mon 2/17	Tue 2/18	Wed 2/19	Thu 2/20	Fri 2/21
Holiday President's Day	**Lecture (11am)** Monday Schedule	**Lecture**	**Lecture**	**Recitation** ASTs and tips for Phase 2
		Due: Team Submission	**Due:** Mini-quiz, weekly check-in	**Due: Project phase 1**

Announcements and updates

Parser generator demo ←

Why not use a parser generator?

- Generally ***slower*** than a hand written recursive descent parser
- May be difficult to specify correctly for more complicated languages
  - Teams have successfully used modern parser generators to specify Decaf, which is relatively simple

Why use a parser generator?

- Generates **“correct”** code
- Good for **prototyping**
- Lexer and parser errors **come for free**
- No need to deal with a hacked parse tree: directly go to the **abstract syntax tree**

ANTLR4

- “**AN**other **T**ool for **L**anguage **R**ecognition”
- Industry standard for parser generation
- LL(*) - Parses **l**eft-to-right, **l**eftmost derivation, infinite* lookahead, adaptive resolution of left recursion
- One grammar, many language targets

List of ANTLR4 Targets

- **Java** (and other JVM languages like **Scala**)
- **TypeScript**
- Go
- C++
- C#
- **Python** (we'll use this for our demo today)
- Swift
- PHP
- Dart

If you use Rust... use Tree-sitter

Parser generator demo

Code available at:

https://github.com/6110-sp25/recitation3