Parsing Lab – Parser Generators

Lab 3: Parsing Expression Grammars

UNIVERSITÄT
LEIPZIG

Task

Build PEG parser for (reasonable subset of) some programming language

- Input is (unannotated) source code
- Output should be parse tree
  (make it an actual tree, but no other semantic actions needed)
- Error handling (or code generation) is not required, but appreciated

Task

Build PEG parser for (reasonable subset of) some programming language

- Input is (unannotated) source code
- Output should be parse tree
  (make it an actual tree, but no other semantic actions needed)
- Error handling (or code generation) is not required, but appreciated

Test your parser

Document your development

- Document subset selected
- Document decisions on tokenizer (lexer) if used
  (sometimes integrated in parser)

Minimal suitable programming language

- While programs with line & block comments (make some up)
- Make it user-friendly
  (allow arbitrary white space between keywords, symbols, etc.)
- Equivalent program should be reconstructable from parse tree
  (do not make literals or identifiers anonymous)

Definition (While program)

While program is either

- Assignment $x\_i = x\_\ell + z$ for $i, \ell \in \mathbb{N}_+$ and $z \in \mathbb{Z}$
  (second summand can be negative)
- Sequence $P\ ;\ P'$ for While programs P and P'
- While loop $\mathrm{WHILE}(x\_i\ !{=}\ 0)\{P\}$ for While program P and $i \in \mathbb{N}_+$
- No other While programs exist

Documentation

- Identify subset (CFG or textual description)
- Document non-standard development choices
  (which lexer generator, which parser generator, etc.)

Submission

- Submit source and (short) documentation in Moodle
- Submission will only be graded “pass” or “fail”
  (parser works as advertised, documentation legible)

Deadline: End of Semester