Compiler Construction Crash Course

Software Evolution — L1P1

Dr. Vadim Zaytsev aka @grammarware, February 2021

UNIVERSITY OF TWENTE. | Formal Methods & Tools

UNIVERSITY OF TWENTE.

Language Processors [1/2]

- Compiler
  - usually text-to-code
- Interpreter
  - directly text-to-result
- Parser
  - text-to-model
- Formatter/Pretty-printer
  - model-to-text

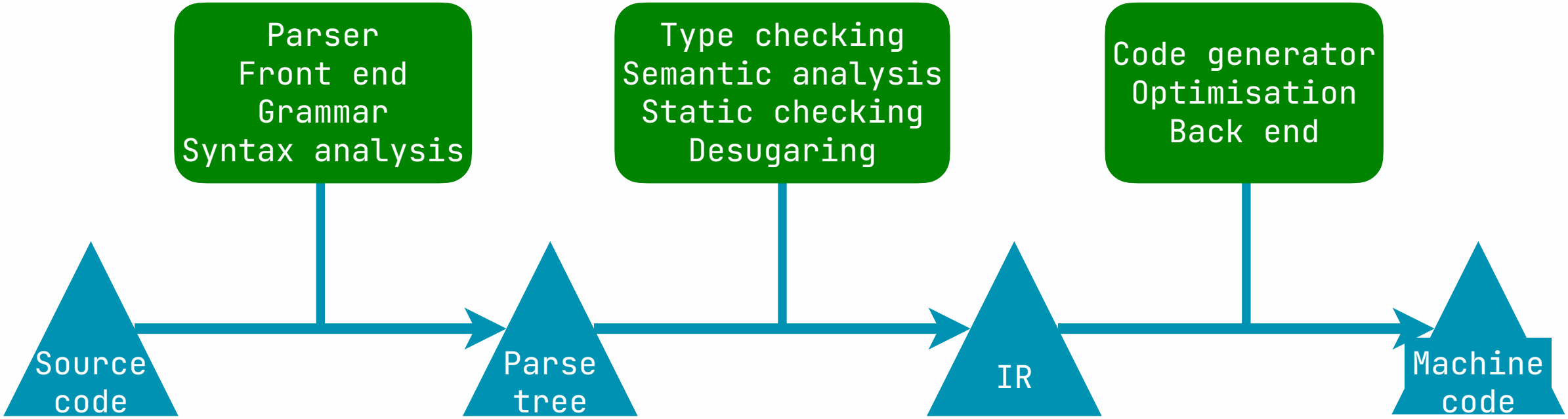
R.Lämmel, *Software Languages*, Springer, 2018.

Language Processors [2/2]

- Preprocessor
  - text-to-text interpreter
- Translator/transformer
  - model-to-model
- Analyser
  - model-to-property
- Generator
  - text-to-artefact

R.Lämmel, *Software Languages*, Springer, 2018.

Compilation Pipeline



Top Choices for This Course [1/3]

- Full compiler
  - text→model→code
- Challenging
- Architecture is known
- Good frameworks are available
  - LLVM, Rascal, Spoofax, …
- All features are open

https://en.wikipedia.org/wiki/Compiler

Top Choices for This Course [2/3]

- Interpreter
  - text→model→execution
- Comparable in difficulty
- A lot of support
  - Language workbenches: Rascal, Spoofax, …
  - Parser generators: ANTLR, …
- Some features are easier
- Some features are harder

https://en.wikipedia.org/wiki/Interpreter_(computing)

Top Choices for This Course [3/3]

- REPL
  - read→eval→print
- Comparable in difficulty
- Architecture is known
- Largely the same as any interpreter
- More interactive/live fun
- Some features are easier
- Some features are impossible (?)

https://en.wikipedia.org/wiki/Read-eval-print_loop

Conclusion

- Language processors aka grammarware
- Pipeline: parser, analyser, generator
- Learn compiler construction *for real*
- Hone your skills aiming for *legacy*
- Compiler? Interpreter? REPL?
- Q&A Sessions @ Canvas
  - ⇒ v.zaytsev@utwente.nl
  - ⇒ https://discord.gg/n7VQAPNBPD