dOvs Eksamens Noter

Hugh Benjamin Zachariae

January 2020

Contents

1 Compiler intro

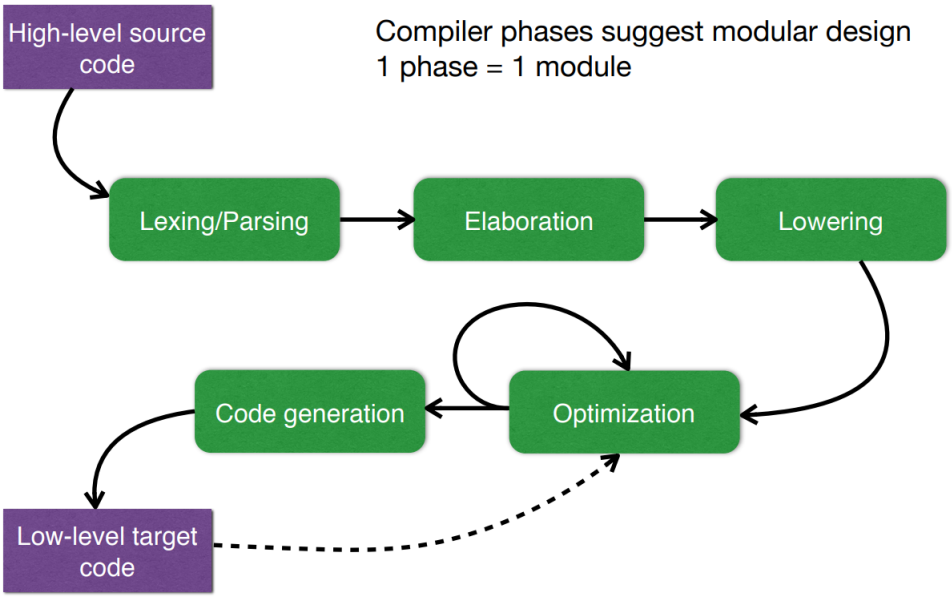

Figure 1: Compiler modular phases.

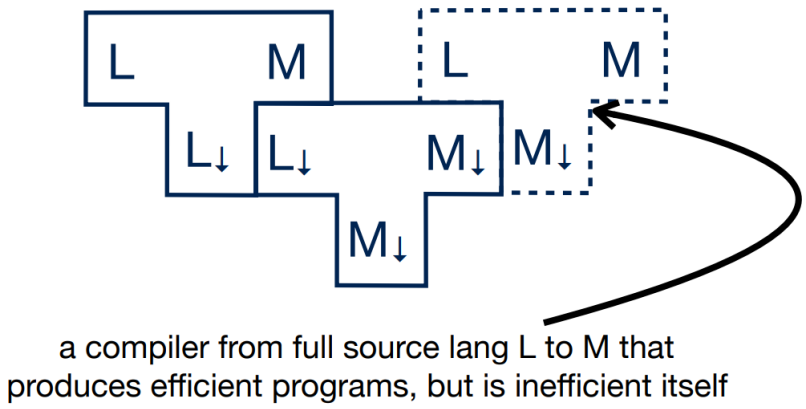

Figure 2: Bootstrap compiling

- **Lexing/Parsing**: String $\rightarrow_{lexing}$ Tokens $\rightarrow_{parsing}$ Abstract Syntax Tree (AST)
- **Elaboration**: *Resolving scope* and *Type checking*. Most errors found here.
- **Lowering**: High-level features to target-language like constructs (e.g. assembly-like). *Intermediate representation*, LLVM.
- **Optimization**: Detect and rewrite expensive operations. Lifting invariants out of loops, parallelization.
- **Code generation**: fx LLVM to X86 (registers, instruction etc.)

- **Bootstrapping compilers**: Compile your language in your own language.

2 Lexical

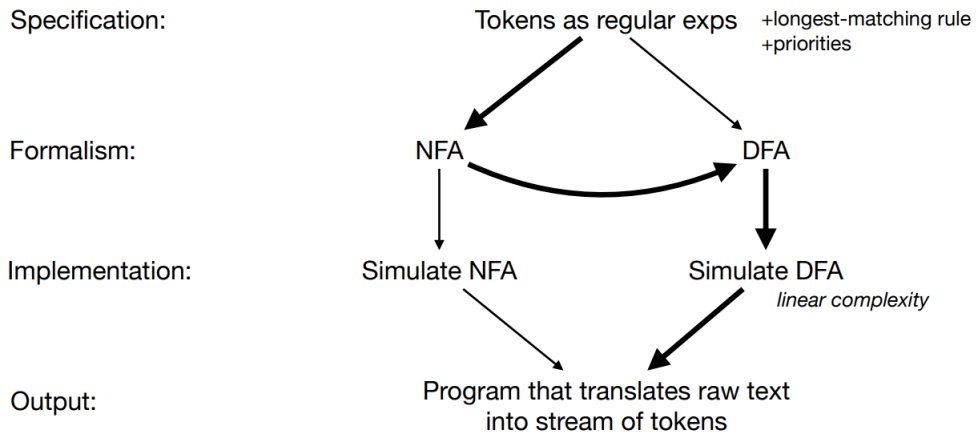

Figure 3: REG to NFA to DFA

- Tokens: E.g. ID("a"), INT, IF etc. Some tokens include metadata like names in ID.
- Non-tokens: comments, whitespace etc.
- REG → NFA → (closures) DFA → Minimized DFA (more effective)
- REG: Handle priorities and longest matching string token wins.
- Ocamllex: Lexer generator