Compilation project

M1 Informatique / M1 Mosig

Gwenaël Delaval Maxime Lesourd

2023–2024

Objectives

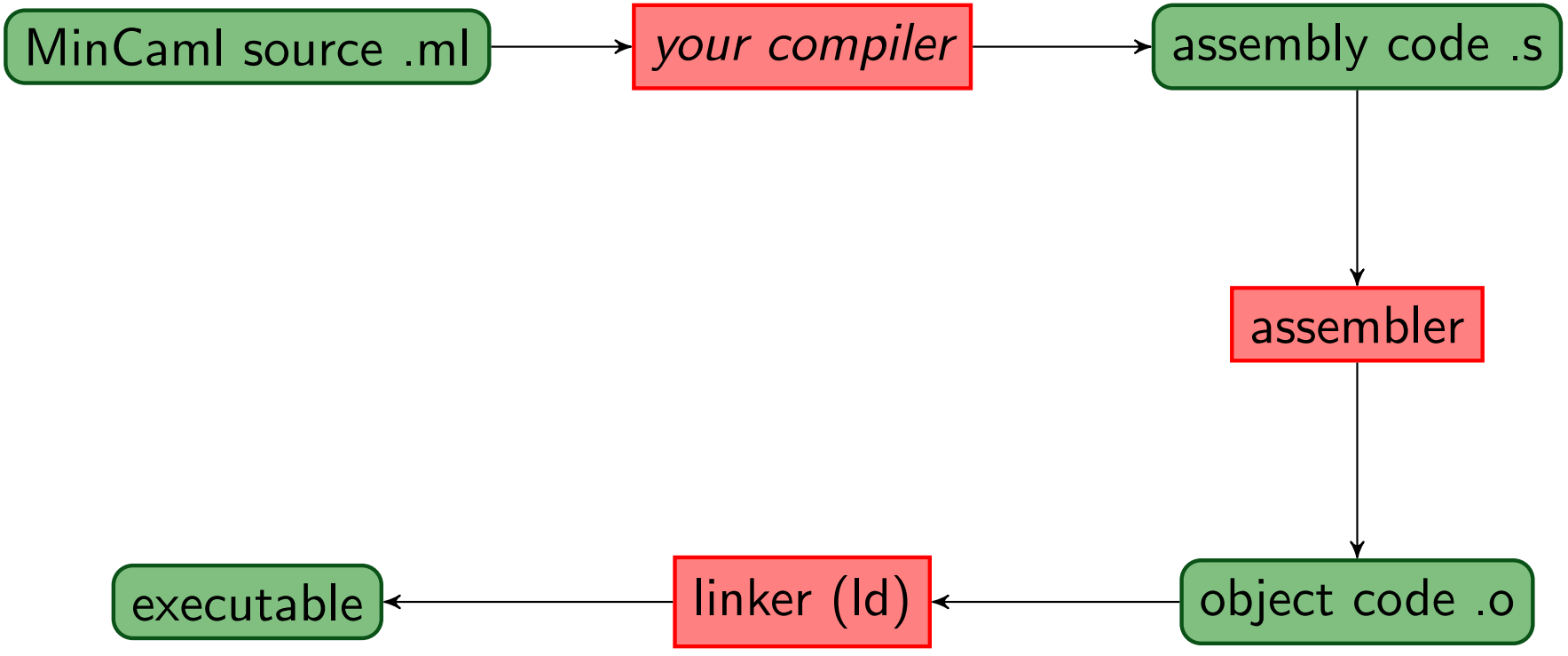
- Compiler programming
  - Writing a compiler for a mini-language (MinCaml, subset of OCaml)
  - Goal : assembly generation
  - Using generators for lexical analysis and parsing
  - Understanding how computations are translated by machines
- Software engineering
  - Write a robust software
  - Understanding and fulfilling specifications
  - Team work
  - Using modern development methods
  - Using versioning (Git)
  - Tests and evaluations

Organization

- 4 week project
- Autonomous team work
- Weekly reports to tutors
- Monitoring and assistance from tutors

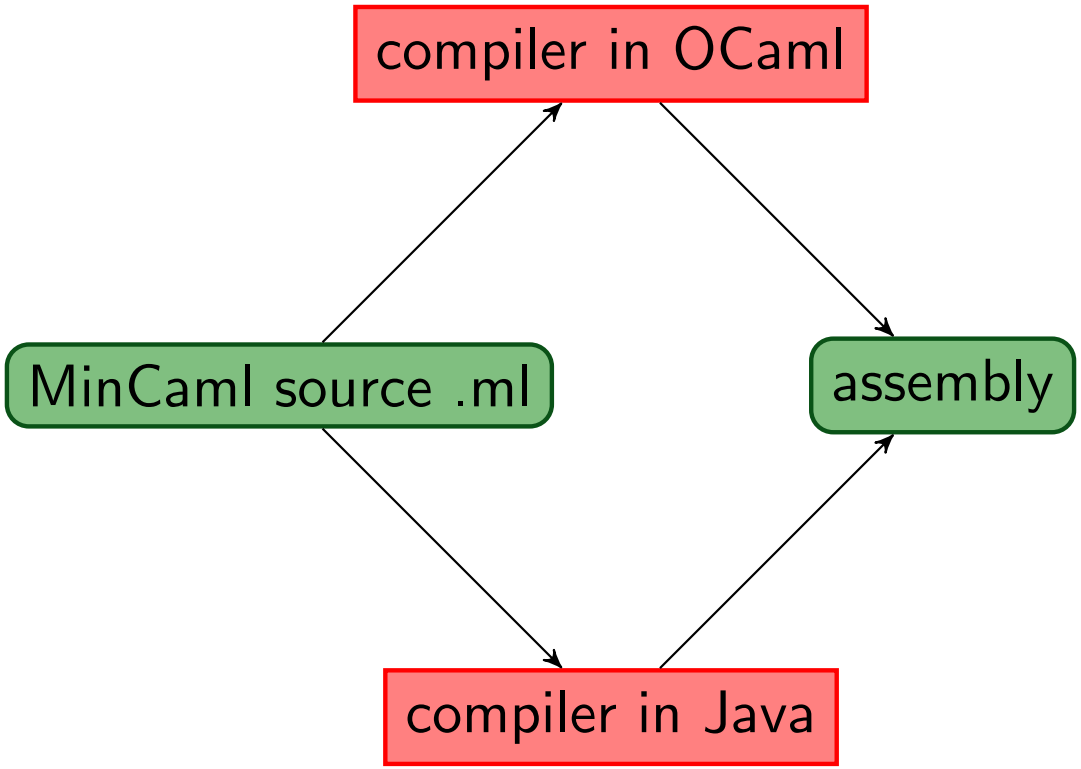
The MinCaml compiler toolchain

From the MinCaml programming language to assembly code, and so on...



What is provided

Skeleton code in OCaml and Java, with lexer/parser and minimal tests



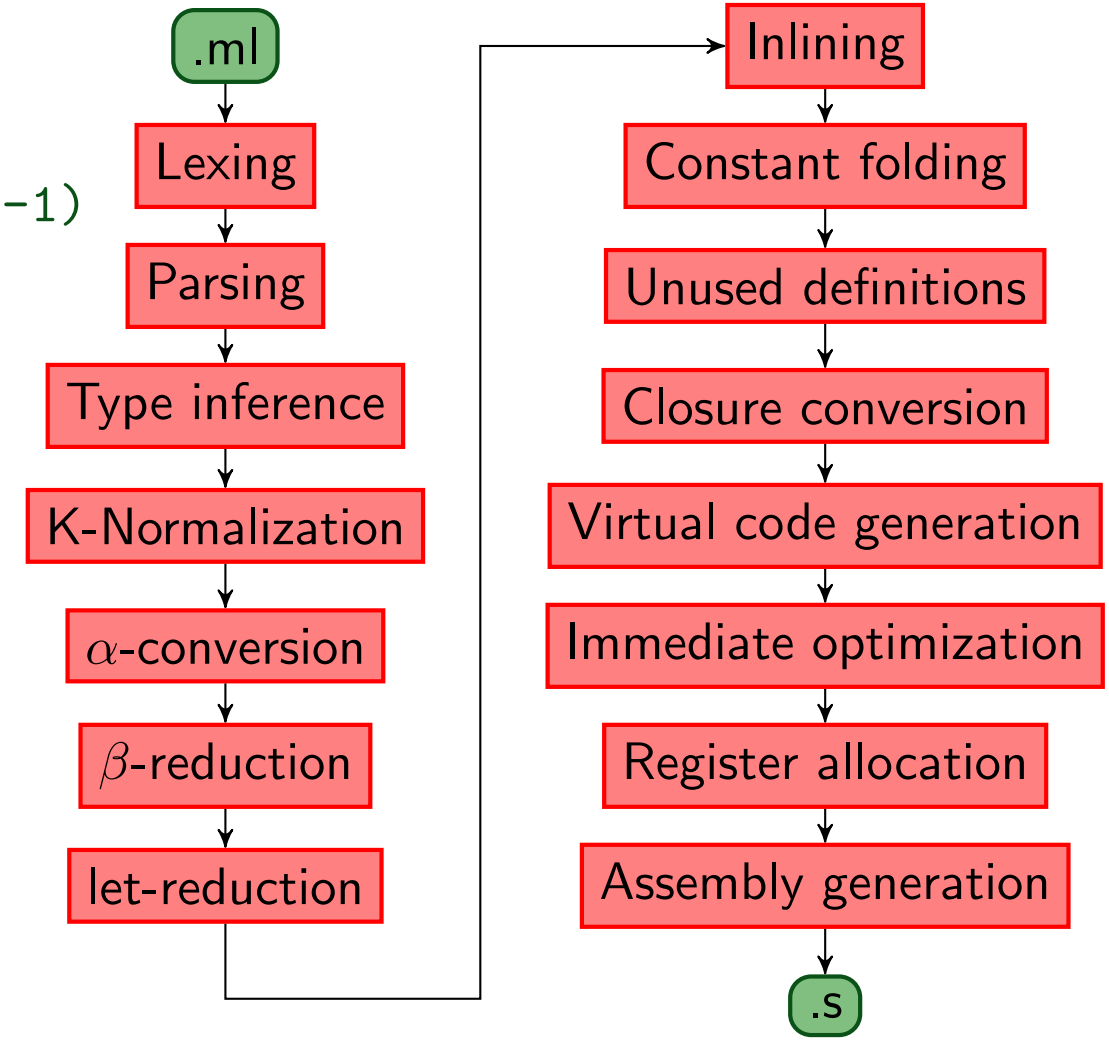
Compiler passes

Material and support provided for guidance

`http://esumii.github.io/min-caml/paper.pdf`

```
let rec fact x =
  if x = 0 then 1
           else x * fact (x-1)
and apply f x = f x in
apply fact 3
```

- A program is an expression
- Type annotations are optional and inferred by the compiler
- A function can have another function as parameter



Why MinCaml ?

- Simple language but expressive enough
  - Adapted for a short project
  - Interesting compiling problems (optimizations, closures)
  - Generated code efficiency comparable to reference compilers
- Functional programming
  - Used more and more in modern programming frameworks
  - Functional features included in core programming languages : C++0x11, Java 1.8, Ruby, Python, Go, Swift...
- Numerous possible extensions : garbage collector, polymorphism, pattern matching, ...

What links with SLPC/PLCD ?

- In SLPC/PLCD : compilation of an imperative language (While)
- Functional language : additional features (higher-order, polymorphism,...) and compilation concepts (α- and β-conversion, closures)
- Typing vs type inferrence (strong type checking without type annotations)
- Different optimization phases

$\longrightarrow$ not strict application of methods seen in SLPC/PLCD, but opportunity to learn more concepts and methods !