Languages, Compilers and Interpreters (Lab)

Lorenzo Ceragioli

September 26, 2024

IMT Lucca

About Me

Teacher: Lorenzo Ceragioli

- Assistant Professor at IMT Lucca (RTD-A)
- Part of the SySMA research group
- Bachelor, Master, and PhD in Computer Science at UniPi :)

Research

- Verification for Quantum Computing and Communication (with Fabio Gadducci, Giuseppe Lomurno and Gabriele Tedeschi)
- Language Based Security (with Pierpaolo Degano, Letterio Galletta and David Basin)
- Non-standard Logic for modeling and verification (with Pierpaolo Degano, Letterio Galletta and Luca Vigano')

Course

Class Schedule: Thursday 11:00 - 13:00 in Fib I

Office hours:

e-mail me at lorenzo.ceragioli@imtlucca.it to set up a meeting

Course page:

lceragioli.github.io/pages/CourseLanguagesCompilersInterpreters.html

Bring your laptop!

Course Syllabus

- OCaml basics
- Implementing the semantics of (simple) programming languages
- Lexing and Parsing with ocamllex and menhir
- Defining and enforcing a type system for a functional language
- Compiling a simple imperative language
- Implementing a simple data-flow analysis tool
- Register allocation

Project

Not surprisingly, you will have to implement:

- An interpreter for a typed minimal functional language
- An interpreter and a compiler for a minimal imperative language, with some data-flow analysis and mild optimization

You must submit: the code and a report containing:

- A discussion on the code motivating the choices and explaining how you have solved the specific problems (it will be clear later)
- A guide on how to run the code

There will be an oral exam based on your submission

The Idea

During a typical lesson:

- We target a fragment of the project
- I show you some tools and approaches for solving a problem or performing a task that is needed for the fragment
- I give you the exact task for the fragment (defining parameters etc)
- I will leave some detail unspecified: your task is not just to translate my solution into ocaml!
- You can start working in class (probably one hour)

Advices

- Keep the code as simple as possible, at least at the beginning
- Learn and use standard libraries
- Take some time for learning OCaml
- Have a precise plan before writing the code
- If you have doubts use the resources in bibliography
- Practice without theory is doomed to failure
- Both code quality (generality, modularity, clarity etc) and efficiency will be considered for the final score, but the quality is more important (and if the code is really modular you can target efficiency at a later time)
- You can skip some tasks and still get 18
  (grades goes from 18 to 30L)