Languages, Compilers and Interpreters (Lab)

2022/2023

Presentation

Instructor: Dr. Letterio Galletta

- Assistant Professor @ IMT Lucca
- Instructor @ UniFI for the course "Distributed Programming for IoT" (MSc. Degree in Computer Science)
- Coach of C3T team @ cyberchallenge.it (born2scan)

Research interest: PL, Language-based security, Program analysis and verification, Blockchain, IoT

- letterio.galletta@imtlucca.it
- http://sysma.imtlucca.it

SCHOOL
FOR ADVANCED
STUDIES
LUCCA

SySMA - Systems Security Modelling and Analysis

The Systems Security Modelling and Analysis (SySMA) unit at IMT Lucca deals with the development of languages and techniques for the analysis, evaluation and verification of possibly distributed systems. The SySMA unit goal is to push the use of formal methods as methodological and automatic tools for the development of high-quality, correct software and systems that are secure, fast, usable, reusable, maintainable, and modular. We also study algorithms and systems to protect the security and integrity of computer systems, the information they store, and the people who use them. We make large usage of formal methods as enabling technology also for the security-by-design development model.
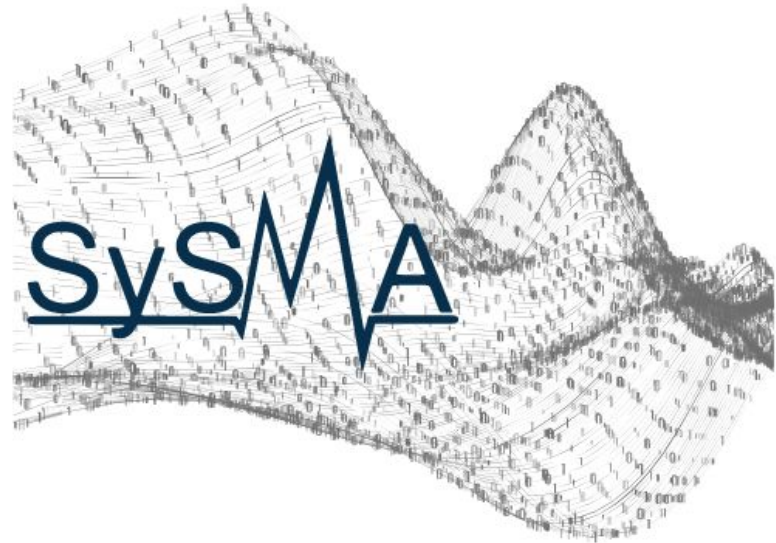
Course Logistics

Class Schedule:

- Monday: 14:00 - 16:00   classroom: O

Office hours: e-mail me to set up a virtual meeting

Course page: https://github.com/lillo/compiler-course-unipi

There will be programming exercises: bring your laptop

Course Readings

Programming tutorials, papers & further material

References will be given during the classes and will posted on the course page

A note about slides

- Slides are made to be used **by the teacher** during the lectures, **not** to be studied **by students**
- Slides will be available on the page course
- Students must study the corresponding chapters of the books and the other referenced material

This Lab Will Teach You

How to implement a small compiler and interpreter

Side effects:

1. The basics of OCaml
2. How to implement parsers using parsers generators
3. How to implement semantic analyses
4. How to use the LLVM toolchain for generating and optimize code

Our main goal: the microc compiler

MicroC is a sublanguage of C with many simplifications compared to real C

- datatypes: only int and char variables, arrays, and pointers
- no structs, unions, doubles, function pointers, ...
- no initializers in variable declarations
- Functions can return only int, char, void, bool
- No pointer arithmetic
- Pointers and arrays are not interchangeable
- No dynamic allocation of memory

```c
/* Function to reverse arr[] from start to end*/
void reverseArray(int arr[], int start, int end)
{
  int temp;
  while (start < end)
  {
    temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
    start = start + 1;
    end = end - 1;
  }
}
```

Our main goal: the MicroC compiler

At the end of each block of lectures there will be an assignment concerning the implementation of a piece of the compiler

Program of the Course

1. Introduction to OCaml
2. Lexing and parsing using ocamllex and menhir
3. Semantic analysis implementation: type checking, scope management, control-flow analysis
4. Introduction to LLVM infrastructure and LLVM intermediate language
5. Code generation

Questions?