Programming Language Semantics and Compiler Design / Sémantique des Langages de Programmation et Compilation

Preamble

Yliès Falcone
ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr
Univ. Grenoble Alpes, and LIG-Inria team CORSE

Master of Sciences in Informatics at Grenoble (MoSIG)
Master 1 info

Univ. Grenoble Alpes - UFR IM^{2}AG
www.univ-grenoble-alpes.fr - im2ag.univ-grenoble-alpes.fr

Academic Year 2017 - 2018

Some practical information

6 ECTS (60 hours).

Lecture sessions: 2 $\times$ 90 min / week

- Yliès Falcone (international + french parcours)
- Laurent Mounier (code generation)
- Henri-Pierre Charles and Fabrice Rastello (guest lectures)

Exercise sessions: 2 $\times$ 90 min / week

- Fabienne Carrier (french parcours - Group 1)
- Gwenaël Delaval (international parcours - Group 2)
- Yliès Falcone (international parcours) - Group 1
- Laurent Mounier (french parcours - Group 2)

Emails: `FirstName.LastName@univ-grenoble-alpes.fr`

Office locations: CEA Minatec (YF), IMAG building (FC, GD, YF, LM).

Meetings are possible (on appointment).

Assessment

Final Exam (FE)

- coefficient: 1.4
- dates: between the 4th and the 8th of December
- 3 hours

Mid-term exam (ME)

- coefficient: 0.3
- program: everything we have seen before the exam
- dates: TBA
- 1.5 hours

Homework or programming project (H)

- coefficient: 0.3
- dates: October-November

$$\text{Finale Grade} = \frac{1.4 \times \text{FE} + 0.3 \times \text{ME} + 0.3 \times \text{H}}{2}$$

References

Pedagogical Resources

All pedagogical resources are on the Moodle:

http://imag-moodle.e.ujf-grenoble.fr/

A. Aho, R. Sethi and J. Ullman

Compilers: Principles, techniques and tools

InterEditions, 1989

H. R. Nielson and F. Nielson.

Semantics with Applications: An Appetizer.

Springer, March 2007. ISBN 978-1-84628-691-9

W. Waite and G. Goos.

Compiler Construction

Springer Verlag, 1984

R. Wilhelm and D. Maurer.

Compilers - Theory, construction, generation

Masson 1994

Compilers: what you surely already know...

A compiler is a *language processor*: it transforms a program:

- from a language we can understand: the programming language,
- to a language the machine can understand: the target language.

Global objectives of the course

- Programming languages, and their description:
  - syntax,
  - semantics.
- General compiler architecture.
- Some more detailed compiler techniques.

Basic objective

Study the translation performed by a compiler:

- the questions raised by the translation;
- the expected properties of this translation;
- how to perform this translation.

The algorithms and design principles used in compilers are general, generic, and are used in many domains of computer science and ICT.

Many programming language paradigms . . .

Imperative languages

- *ex: FORTRAN, Algol-xx, Pascal, C, Ada, Java*
- notions: control structure, (explicit) memory assignment, expressions, types, . . .

Functional languages

- ex: *ML, CAML, LISP, Scheme, etc*
- notions: term reduction, function evaluation, recursion, . . .

Object-oriented languages

- ex: *Java, Ada, Eiffel, C++*
- notions: objects, classes, types, inheritance, polymorphism, . . .

Logical languages

- ex: *Prolog*
- notions: resolution, unification, predicate calculus, . . .

Web languages

- ex: *JavaScript*, *PHP*, *HTML*
- notions: scripts, markers, . . .

etc.

. . . and many architectures to target!

- Complex instruction set computer (CISC)
- Reduced instruction set computer (RISC)
- VLIW, multi-processor architectures
- dedicated processors (DSP, . . .)
- embedded systems (mobile phones, . . .).
- etc.

We will mainly focus on:

Imperative languages

- data structures
  - basic types (integers, characters, pointers, etc)
  - user-defined types (enumeration, unions, arrays, ...)
- control structures
  - assignments
  - iterations, conditionals, sequence
  - nested blocks, sub-programs

"Standard" general-purpose machine architecture: (e.g. ARM, iX86)

- heap, stack and registers
- arithmetic and logical binary operations
- conditional branches

Course programme (overview)

1. Introduction, architecture of a compiler
2. Static semantics of a language and type analysis
3. Natural operational semantics
4. Structural operational semantics
5. Provably-correct implementation (of a compiler for a simple machine)
6. Axiomatic semantics
7. Denotational semantics
8. Intermediate-code generation
9. Code optimization
10. Machine-code generation
11. Dynamic Compilation and compilation for embedded systems (Henri-Pierre Charles CEA, Grenoble)
12. Performance considerations and loop optimization (Fabrice Rastello, Inria, Grenoble)
13. Security aspects (Laurent Mounier)