II.3523 – Formal Approaches, Languages and Compilers

General Information

Title : Formal Approaches, Languages and Compilers
Code of the module : II.3523
Person in charge : Ammar KHEIRBEK
ECTS : 5
Average amount of work hours : 100 à 150h (42h face to face)
Teamwork : yes
Keywords : Finite state machine, Stacked state machine, Turing machine, Regular expressions, Regular languages, Context-free languages, Compilation steps, Lexical analysis, FLEX, Syntax analysis, BISON, Parser, Semantic analysis, Abstract syntax tree, Code generation.

Présentation

This module introduces the basic Computation Models that are used in many areas of Computer Science and Automation as an essential tool for modelling of automatically controlled systems (such as drones, airplanes, or other medical devices) and autonomous systems (such as vehicles, subways, and trains) widely implemented in the daily life of our societies.
The module also focuses on the direct relationship between these models and the domain of formal languages and the grammars needed to formulate these languages. The theoretical foundations of programming language design will be studied, as well as the operational creation of these languages.
These formal languages and grammars can also be used in several areas of Computer Science, such as natural language processing, information retrieval, knowledge management, etc.
Finally, a more in-depth study of compilation techniques will be carried out to enable students to build a compiler from scratch. Compilers are software programs that take a program written in a given language and generate code that can be directly executed by the machine. Compilers are considered to be some of the most advanced software that students will be able to design and implement during their studies. This study will be used directly by the students in a semester project: to develop a complete compiler that takes the Markdown markup language as its source, and produces the HTML markup language as its object.

Pedagogical Objectives

- Modelling of systems using computational models
- Choice of the programming language(s) best suited to a given development context
- Define a BNF grammar
- Use parser generators
- Implement an interpreter

Prerequisites

A good knowledge of a programming language is essential (ideally C, C++ or Java). A background in basic mathematics is desirable, especially in general algebra (set theory). A markup language such as HTML for labs.

Content/programme

Concepts

Computational models	Formal Languages	Compilation
Finite state machine : Deterministic & non-deterministic Regular Expressions	Classification of Chomsky Regular Languages	Stages of compilation Lexical Analysis
Stack Automaton Proofs Systems	Context-free Langages Context-free grammars LL (1), LR (1), LALR (1)	Syntax Analysis, Abstract syntax tree Semantic analysis Type checking Code generation
Turing Machine	Enumerated recursive languages Decidability	

Tools

For the compilation project (Markdown → HTML) students will use a lexical analyzer (FLEX, ver. 2.6), and a parser (BISON, ver. 3.8.).

PEDAGOGICAL OBJECTIVES

Learning methods

14 sessions of 3 hours each, comprising:

- 28 hours of theory classes
- 12 lab hours for the compilation project (project throughout the semester)
- 2 hours of quizz

Assessment methods

All assessments are individual.

- Activities and tutorials: 10% of the total
- Quizzes: 20% of the total
- Semester project (3 students per project): 30%.
- Final exam: 40%.

Working language

Module delivered entirely in English. Students' deliverables can be in French or English (preferred).

BIBLIOGRAPHY, WEBOGRAPHY, OTHER RESOURCES

- Tom Stuart. Understanding Computation: From Simple Machines to Impossible Programs. O'Reilly Media, Inc. 2013.
- Alfred Aho , Monica Lam, Ravi Sethi , Jeffrey Ullman. Compilers: Principles, Techniques, and Tools (Dragon Book). Addison Wesley; 2nd edition, 2006.

- John Hopcroft, Rajeev Motwani, Jeffrey Ullman. Introduction to Automata Theory, Languages, and Computation. Pearson; 3rd edition, 2006.
- Ammar Kheirbek. Introduction to Automata and Formal Languages. Publications of Damascus University. 2nd edition, 2006.
- Guy L. Steele. Growing a language. Higher-Order and Symbolic Computation, 12(3), 221-236. 1999.
- John E. Savage. Models of Computation: Exploring the Power of Computing. Addison-Wesley. 1998.
- Many Online Sources.