Module Description, available in: EN

Theoretical Computer Science

General Information

Number of ECTS Credits

3

Module code

FTP_TheoComp

Valid for academic year

2024-25

Last modification

2023-09-26

Coordinator of the module

Olivier Biberstein (BFH, olivier.biberstein@bfh.ch)

Explanations regarding the language definitions for each location:

- Instruction is given in the language defined below for each location/each time the module is held.
- Documentation is available in the languages defined below. Where documents are in several languages, the percentage distribution is shown (100% = all the documentation).
- The examination is available 100% in the languages shown for each location/each time it is held.

	Lausanne			Lugano	Zurich		
Instruction					**X** E 100%		
Documentation					**X** E 100%		
Examination					**X** E 100%		

Module Category

FTP Fundamental theoretical principles

Profile Recommendations

This module is recommended for the following profiles:

	Zurich
Profile	CS

Lessons

2 lecture periods and 1 tutorial period per week

Entry level competences

Prerequisites, previous knowledge

Good knowledge of programming, algorithms and discrete mathematics.

Brief course description of module objectives and content

The aim of this module is to deepen some basic theoretical aspects of computer science. The master students will learn that ...

- formal languages and automata are essential concepts to describe different types of problems and computations;
- Computability/decidability are central to explain that for many problems seem to have an intuitive solution, although they can not be solved by algorithms;
- Complexity deals with the amount of time required to solve a problem, and there are many very practical problems that can not be solved in reasonable time or space.

Aims, content, methods

Learning objectives and competencies to be acquired

- The students understand that three different mathematical formalisms (finite state automata, regular grammars, regular expressions) are equivalent and define the set of regular languages. Finite state automata and regular expressions are widely used and will be explained using examples from lexical analysis, modeling of simple state-based systems, telecommunications protocols and program verification.
- The students realize that programming languages with regular languages can not be fully described. Context-free grammars, on the other hand, are suitable for developing all modern programming languages. Parsing is closely linked to the context-free languages. Using parser generators, students can explain top-down and bottom-up parsing.
- The students know that many problems are undecidable, i.e. that there are no algorithms to solve them, or rather that not everything is predictable. Such intuitively difficult to understand problems occur e.g. in the case of operating systems (deadlock problem), object-oriented programming languages (subtype decision) or program verification.
- The students understand that decidable problems are classified according to the resources needed to solve them (time or space) and know the major complexity classes (P, NP, EXP, PSPACE), their differences, and interrelationships (e.g., hierarchy).
- The students understand the concept of nondeterminism, which plays an essential role in the study of complexity. The complexity class NP (nondeterministic polynomial time) includes a subclass of very practical problems that are not solvable in reasonable time. Cryptology, machine vision, various optimization problems and many other areas are affected by such problems. Students can demonstrate that such problems are indeed unsolvable in reasonable time, and know some ways to circumvent this limitation through approximation techniques.Many problems known as NP-complete are presented and investigated.

Module content with weighting of different components

The module is divided into three parts:

1. Languages and automata (about 36 %)
    1. Alphabets, words, formal languages, grammars
    2. Finite state automata, regular languages/grammars, regular expressions, nondeterminism
    3. Pushdown automata, context-free languages/grammar
    4. Turing machines
2. Computability/decidability (about 21 %)
    1. Various computation models, Church-Turing thesis
    2. Reduction of a problem to another
    3. Decidable/undecidable problems
    4. Computable/uncomputable functions
3. Complexity (about 43 %)
    1. Types of complexity (time, space)
    2. Complexity classes, polynomial time complexity, NP, polynomial time reductions, NP-completeness
    3. Approximation methods

Teaching and learning methods

- Frontal teaching
- Presentation and discussion of theoretical topics
- Discussion of practical examples to reduce the gap between theory and practice
- Exercises and self-study of selected topics

Literature

Introduction to the Theory of Computation, Michael Sipser, Cengage Learning, 3rd International Edition, 2013.
Reference: http://www-math.mit.edu/~sipser/book.html (Homepage of author of the book)

Computers Ltd.: What They Really Can't Do, David Harel, Oxford University Press, 2000.

Introduction to automata theory, languages, and computation, J.E. Hopcroft and J.D. Ullman, Addison-Wesley Publishing Company, Reading, MA, 1979.

Assessment

Certification requirements

Module does not use certification requirements

Basic principle for exams

As a rule, all standard final exams are conducted in written form. For resit exams, lecturers will communicate the exam format (written/oral) together with the exam schedule.

Standard final exam for a module and written resit exam

Kind of exam

Written exam

Duration of exam

120 minutes

Permissible aids

Aids permitted as specified below:

Permissible electronic aids

Summary (10 pages)

Other permissible aids

No other aids permitted

Special case: Resit exam as oral exam

Kind of exam

Oral exam

Duration of exam

30 minutes

Permissible aids

No aids permitted