Università degli Studi di L'Aquila - Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Course catalogue

Page compiled on 18/11/2019

Programme of Course "Teoria della Calcolabilità e Complessità"

- Code: F0150
- Type of course unit: Compulsory (Bachelor Degree in Computer Science curriculum General)
- Level of course unit: Undergraduate Degrees
- Semester: 2

Number of ects credits: (Bachelor Degree in Computer Science) 6 (workload 150 hours)

Teachers: Filippo Mignosi (Filippo.Mignosi@univaq.it)

1	**Course objectives**	The course deals in the first part with the intrinsic limits of what can be calculated with a computer and in the second part introduces the theory of Complexity. At the end of the lessons, and passing the exam, the student should be able 1) to know the fundamentals of the theory of calculability and complexity and to understand the intrinsic limits to what any calculation model is able to do in terms of acceptance of languages but also in terms of calculable functions and, therefore, to understand the thesis of Church-Turing. 2) To know the main resources available to those who calculate, i.e. time and space, and 3) to know and to understand the first major classes of languages both in the theory of computability and in theory of complexity including those in the randomized case. 4) To know how to distinguish potentially unsolvable problems both in the theory of calculability (undecidability, not calculable or not R.E.) and in the theory of complexity (not belonging to class P) in order to find satisfactory alternatives in practice; therefore 5) to know of the existence of methodologies, not covered in this course but some present as topics in other courses, such as model checking, static analysis, analysis and testing, combinatorial optimization theory, approximate algorithms etc. 6) to be able to model problems and propose efficient algorithmic solutions. 7) To know how to distinguish incomplete demonstrations (proofs) from the complete ones. Being able to follow demonstrations with chains of complex logical deductions and to know how to complete and to integrate proofs. 8) To describe the topics of the course with sufficient mathematical rigor, 9) to describe algorithms in a complete way, even if not totally formal. 10) to formalize and communicate problems, ideas and algorithmic solutions. 11) To develop the skills necessary to undertake studies subsequent with a high degree of autonomy. 12) To know how to learn with ease topics, related to the course, of which there is only partial knowledge.
2	**Course content and learning outcomes (dublin descriptors)**	Topics of the module include: • At the beginning of the course it is given the detailed program of the course, referring to the main textbook. This detailed program is also sent to a mailing list created every year containing the official mail addresses of students that want to participate and to anyone who has the right to request it, such as for non-attending students. Briefly, the program is divided into two parts, as from the name of this teaching, namely Theory of calculability and Theory of complexity, which are interconnected. Indeed some results of the first part find an obvious continuation in the second part. Given a common basis, the second part starts every year slightly after the temporal half of the course. The first part starts with the correspondence between languages ??and problems and the relationship between accepting a language (of n-upe of strings) and calculating functions is discussed in depth. Turing machines are introduced and simple calculation exercises are performed, such as calculating the +1 or +2 in binary. Programming techniques are developed with Turing machines and changes are made to the calculation model, showing its equivalence with numerous variants. We pass from the non-deterministic model (and motivates it with links to cryptography) to the multi-tape model (analogies to parallellism) up to the machines with two counters, always showing equivalence theorems between models. The Church Turing Thesis is developed starting from the previous demonstrations. Moreover we talk about the robustness of the concept of algorithm and computation and we just mention the strong Church-Turing thesis and the quantum model. The classes of the R.E. and of the Recursive languages are introduced, ??the reductions are defined and theorems are developed relatively to the Boolean operations inside above classes and the

		theorems that link the reductions to the previous classes. The languages ??Lu and Ld are defined and their classical properties are proved, and finally the first part is completed with the languages ??Le, Lne and their properties and the Rice theorem. In the second part, we talk about treatable and intractable problems to start the theory of computability and define the classes P and NP. Cryptography is always used to motivate the NP class. Then we introduce polynomial reductions, the class of NP-complete languages ??and develop connected theorems concerning these arguments. The NP membership of the SAT problem is proved and polynomial reductions are developed among the several complete NP problems discussed in the book in order to familiarize students with the class of NP-complete problems. Then we pass to the class of co-NP, PS, NPS and connected theorems such as the Savitch theorem. finally we move on to the classes of languages ??related to the randomization i.e. ZPP, RP, Co-RP and are studied relationships between them and the previous classes. The course ends with the description of a randomized primality test. On successful completion of this module, the student should : • The course deals in the first part with the intrinsic limits of what can be calculated with a computer and in the second part introduces the theory of Complexity. At the end of the lessons, and passing the exam, the student should be able 1) to know the fundamentals of the theory of calculability and complexity and to understand the intrinsic limits to what any calculation model is able to do in terms of acceptance of languages but also in terms of calculable functions and, therefore, to understand the thesis of Church-Turing. 2) To know the main resources available to those who calculate, i.e. time and space, and 3) to know and to understand the first major classes of languages both in the theory of computability and in theory of complexity including those in the randomized case. 4) To know how to distinguish potentially unsolvable problems both in the theory of calculability (undecidability, not calculable or not R.E.) and in the theory of complexity (not belonging to class P) in order to find satisfactory alternatives in practice; therefore 5) to know of the existence of methodologies, not covered in this course but some present as topics in other courses, such as model checking, static analysis, analysis and testing, combinatorial optimization theory, approximate algorithms etc. 6) to be able to model problems and propose efficient algorithmic solutions. 7) To know how to distinguish incomplete demonstrations (proofs) from the complete ones. Being able to follow demonstrations with chains of complex logical deductions and to know how to complete and to integrate proofs. 8) To describe the topics of the course with sufficient mathematical rigor, 9) to describe algorithms in a complete way, even if not totally formal. 10) to formalize and communicate problems, ideas and algorithmic solutions. 11) To develop the skills necessary to undertake studies subsequent with a high degree of autonomy. 12) To know how to learn with ease topics, related to the course, of which there is only partial knowledge.
3	**Course prerequisites**	Mandatory: to have developed and implemented basic algorithms in any programming language. To know the initial data structures such as queues and stacks and to know how to use elementary forms of recursion and to know well the visits in amplitude and in depth of graphs and trees. Prerequisites: algorithms and data structures with laboratory
4	**Teaching methodsand language**	The lessons are usually frontal and few self-assessment “in itinere” tests are carried out which are however completed by frontal explanations. Exercises are done “in class” but their amount is the minimum for developing the ability to apply the acquired knowledge. An attempt is made to make the teaching centered on the students by asking to attending students what do they expect in relation to the course, contents and assessment methods. The creation of discussion groups is stimulated both with and without the presence of the teacher regarding the subject of study, and the cross-discussion is stimulated. Students are encouraged to ask questions and develop ability to criticize. Among the teaching methods, motivational techniques such as references to cryptography or to recent news events to arouse interest are also used. The modalities of exams are seen as a didactic method, and indeed, to generate attention, on almost every topics it is explained what are the modalities of exams on that specific topic. **Language**: Italian

		Reference textbooks • Hopcroft, Motwani, Ullman, ***AUTOMI LINGUAGGI E CALCOLABILITA'*** . Pearson, Addison Wesley. • Arora, Barak, ***Computational Complexity: a modern approach***. Cambridge University press.
5	**Assessment methods**	A written test is made which is called "interactive". Indeed after the student answered to first questions regarding concepts and basic definitions and tests necessary to pass the exam, the correction is done together with the student himself and, as a general rule, questions are asked according to the answers given, to the precision, to the correctness, to the exposition capacity of the student, and according to the demonstrated logical capacity. This occurs in several similar phases until the commission reaches a judgment deemed valid and reliable. In general, a student must know the basic definitions and basic techniques to pass the exam and then the grade will also grow in proportion to the subjects in which he has been able to answer and also, and we repeat, depending on the answers given. , or rather from the precision, correctness and exposition capacity of the student, from the demonstrated logical capacity. In particular, to assess the ability to apply knowledge and understanding and also the learning skills, students' ability to understand and integrate demonstrations and logical reasoning on 1) topics related to course but which they were not strictly treated as topics of the course or discussion on the most advanced topics in the case of students with an evaluation close to the maximum or 2) topics in which the students expressed uncertainties or inaccuracies to see if they can be more certain or precise. Sometimes a student's self-assessment is also asked and this is always followed by a discussion.