Computer Science—Algorithms, Languages and Logic
M.Sc., 120 cr, 2 years

Peter Damaschke

2011-12-01

1 Programme aim

Version 2011-12-01 by Peter Damaschke.

Computer systems are becoming increasingly powerful and intelligent, and they rely on increasingly sophisticated techniques. To master the complexity of these systems it is essential to understand the core areas of computer science. While computer systems and their applications develop rapidly, the underlying foundations of computing are mature mathematical theories.

This programme offers a comprehensive foundation in the science of programming and computing, thus providing lasting knowledge in the field. It gives the student a strong basis for developing the computer applications of today and tomorrow and for conducting innovative research and promoting development.

After completion of their studies, the students will be able to

- apply solid knowledge in the mathematical and logical foundations of computing, to computational problems appearing, e.g., in industry and the public sector, thereby taking into account aspects like tractability, complexity, correctness, and security,
- create models of real-world scenarios that are both meaningful and amenable to analysis, thereby choosing the right level of abstraction,
- develop correct and efficient computer programmes and systems that satisfy given requirements under given constraints,
- analyse and test systems, evaluate, predict, prove and verify their essential properties,
- identify and handle complex computational problems,
- communicate their ideas and results to both specialists and nonspecialists, give structured and scholarly presentations in oral and written form,
- find scientific information and integrate knowledge, identify their own needs for gathering further knowledge and do the necessary self-study,
- work in project groups and international environments, take a leading role,
- critically judge systems, results, and the use of information technology also from a social point of view, and be aware that results of computations crucially depend on model assumptions.

2 Who should apply

The programme is intended for students who wish to study the core areas of computer science on an advanced level in order to prepare themselves for research and development particularly in the software industry. It also provides an ideal basis for academic research in computer science.

Most students will have a BSc in computer science. However, the programme can also serve as a conversion course for students with BSc in related subjects, such as mathematics, physics or engineering sciences, provided they have basic knowledge of mathematics and programming, and have completed an introductory computer science course such as data structures or algorithms.

3 Why apply

Students will obtain a strong computer science background and thus gain access to a wide range

of opportunities in the information technology industry. Students acquire generic skills and lasting subject knowledge and are in a good position to understand and contribute to technological advances. Products and whole companies are based on advanced algorithmic techniques, and these industries employ skilled computer scientists. The programme also provides the students with an excellent background for future PhD studies in computing, which can lead to a career as an academic researcher or computing teacher.

4 Research connections

Chalmers pursues internationally recognised research, also in cooperation with industry, in all core areas of the programme. Chalmers is well-known for its research in functional programming and played a major role in the design and development of the standard lazy functional language Haskell. Powerful software tools for testing and combinatorial problem solving have widespread use. Methods from programming language theory are applied to problems in security. Pioneering work is conducted in type theory and computer assisted theorem proving. Researchers in programming logic also collaborate with linguists in the field of natural language processing. Another important area is the design and analysis of algorithms and their applications in bioinformatics and networks in a broad sense, and in large-scale optimisation. Teachers are active researchers in these fields, which gives the programme scientific quality and depth.

5 Specific eligibility

5.1 Undergraduate profile

Major in Computer Science, Computer Engineering, Mathematics, Engineering Physics or Information Engineering.

5.2 Prerequisites

Mathematics (including Discrete Mathematics, Linear Algebra, and Analysis), Statistics, Programming, basic knowledge of Data Structures.

6 Programme content

The core of the programme covers four main branches of computer science:

- *Algorithms*, including artificial intelligence, machine learning and optimisation,
- *Logic*, including applications in hardware and software verification,
- *Programming languages*, with underlying principles, implementation techniques and advanced programming techniques,
- *Computer security*, including cryptography and language-based security.

Students have to take the following core courses:

- Algorithms
- Logic in Computer Science
- Programming Language Technology

The optional segment of the programme offers the student a broad range of courses in other areas of computer science, bioinformatics, software engineering, mathematics, and other related fields.

At least 5 of the following profile courses, divided in 4 tracks, must be taken:

- **Algorithms:** Advanced Algorithms, Discrete Optimization, Algorithms for Machine Learning and Inference, Artificial Intelligence
- **Logic and Verification:** Models of Computation, Types for Programs and Proofs, Software Engineering using Formal Methods, Hardware Description and Verification[1]
- **Programming Languages:** Compiler Construction, Advanced Functional Programming, Frontiers of Programming Language Technology, Parallel Functional Programming
- **Security:** (no core course) Cryptography, Programming Language Based Security

[1] Under discussion, may be cancelled.

Students may specialise on a track or aim at a broader education and mix courses from several tracks. The curriculum is filled with 4 elective courses that can be freely chosen and may also come from neighbouring programmes. It is also possible to take an elective project course and PhD level courses.

The programme is completed with a 30 cr master's thesis. The thesis proposal has to be approved by the programme director.

Appendix

The programme has the ambition to offer the students a variety of courses that are tied to the research strengths at the Department of Computer Science and Engineering. It provides freedom of choice, thus enabling individual study plans, but at the same the tracks give guidance for meaningful paths to go. The roles of most of the courses are detailed below.

The **compulsory core courses** represent the three cornerstones of the programme: Algorithms, Logic in Computer Science, Programming Language Technology.

Courses in the **Algorithms track** rely on the basic Algorithms course (or knowledge gained in equivalent courses) and can be studied in any order.

Advanced Algorithms is a generic algorithms course that covers the main techniques of design and analysis of algorithms, but in selected fields it goes more into depth than the basic course. In particular it provides solution methods for computationally hard problems. It adopts a rigorous analytical approach and emphasises correctness, efficiency, and accuracy.

Discrete Optimization, in contrast, focuses on a special but very important class of problems, integer linear programs (ILP), and provides methods for modelling (that is, completely specifying) real-world optimisation problems as ILP, as well as methods for solving them.

Algorithms for Machine Learning and Inference is another specialised course. It shows how algorithms can extract information and automatically draw conclusions from data sets, on solid scientific grounds.

Artificial Intelligence (AI) has similar contents but a unique shape, consisting of a block of introductory lectures followed by projects to be carried out by small groups of students. Logic appears as a formalism to specify and solve problems in AI.

Structural Bioinformatics is not among the profile courses but can be taken as a deepening course in Algorithms: Students with an interest in computational molecular biology learn about problems in this application domain that are accessible to the algorithmic methods seen in the other courses.

The **Logic and Verification** track builds upon the compulsory Logic in Computer Science course. There is also some interaction with Programming Language Technology, thus students will be helped by having done this compulsory course before. However the profile courses can be read in any order.

Models of Computation is a course about formalisms that make the intuitive notion of computability explicit.

Types for Programs and Proofs gives broad knowledge of type systems for programming languages, including interactive programming and proof systems using dependent types.

Software Engineering using Formal Methods shows how to formally express and verify safety properties of programmes. This way it applies logic to verification tools.

The **Programming Languages** track comprises courses on several aspects of programming languages, which is the main tool to make computers actually work.

Compiler Construction addresses the syntactic structure of programming languages, including type systems, and translation of source code into executable code. It can be noticed that problems in run-time organisation, code optimisation, and register allocation build a bridge to the field of discrete optimisation.

Advanced Functional Programming explores the powerful mechanisms that functional programming offers to solve real problems and to structure large programs. A focus is on library design, embedded languages, and types. Specification, program properties, and reasoning about correctness are common themes in this master's programme.

Parallel Functional Programming has been introduced as a course that contributes to the increasingly important area of parallelism (e.g., multi-core processors).

The **Security** track is not directly built upon a core course but has obvious relationships to algorithms and problem complexity as well as programming languages and verification of properties:

Cryptography presents, among other topics, public key primitives based on the computational hardness of mathematical problems, hash functions as parts of cryptographic protocols, and security guarantees.

Programming Language Based Security is based on the idea that many attacks work on the application level, and these are typically specified in programming languages. A direct benefit of language-based security is the ability to naturally express security policies using the well-developed area of programming languages.

The **Project in Computer Science** is not a course in the usual sense, nor does it belong to a track. It gives students the opportunity to do supervised work on a freely chosen subject which, however, must fit in the programme and be nontrivial.