UNIVERSITY OF GOTHENBURG

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DIT233 Types for Programs and Proofs, 7.5 credits

Typer för program och bevis, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by Department of Computer Science and Engineering on 2019-02-08 to be valid from 2019-09-02, autumn semester of 2019.

Field of education: Science 100%
Department: Department of Computer Science and Engineering

Position in the educational system

The course is offered within several programmes. It is also a single subject course at the University of Gothenburg.

The course can be part of the following programmes: 1) Computer Science, Master's Programme (N2COS) and 2) Computer Science, Bachelor's Programme (N1COS)

Main field of studies	*Specialization*
Computer Science	AXX, Second cycle, in-depth level of the course cannot be classified

Entry requirements

To be eligible to the course, the student should have successfully completed 120 credits of studies in computer science or mathematics, or equivalent. Specifically, a successfully completed 7.5 credit course in discrete mathematics (e.g., DIT980 Discrete Mathematics for Computer Scientists, or equivalent) and a successfully completed 7,5 credit course in functional programming (e.g. DIT143 Functional Programming, or equivalent is required.

Applicants must prove knowledge of English: English 6/English B or the equivalent level of an internationally recognized test, for example TOEFL, IELTS.

Learning outcomes

On successful completion of the course the student will be able to:

Knowledge and understanding

- describe several kinds of type systems, their underlying design principles, and their semantic foundation
- read and present a research topic in the area

Competence and skills

- program in a dependently typed functional programming language
- prove theorems in a dependently typed programming language using the propositions-as-types principle
- use deduction formalisms to present type systems and operational semantics of programming languages

Judgement and approach

- critically analyse type systems and prove properties about them

Course content

The development of powerful type systems is an important aspect of modern programming language design. This course provides an introduction to this area. In particular it introduces the notion of dependent type, a type which can depend on (is indexed by) values of another type, for example, the type of vectors indexed by its length. Dependent types are versatile. Through the Curry-Howard identification of proposition and types virtually any property of a program can be expressed using dependent types. The aim of the course is to give a solid and broad foundation in type systems for programming languages, and also give examples of type-based technologies in computer science.

- introduction to lambda calculus and simple type theory
- introduction to operational semantics and type systems
- dependent type theory
- the Curry-Howard identification of propositions as types
- programming in Agda, a proof assistant
- presentation of advanced topics in type systems

Form of teaching
Lectures, exercise sessions, supervision.

Language of instruction: English

Assessment
The course is examined by an oral presentation normally performed in pairs, and by an individual take home examination. In addition, to receive a higher grade than G/Pass the student has to pass an individual oral examination.

The take home exam includes both theoretical problems and programming assignments. For the oral presentation, the student can select either a research paper or a chapter in the course book involving applications of type systems not covered in the lectures.

If the take-home exam is handed in too late it is failed. An oral examination is used as reexamination.

If a student, who has failed the same examined component twice, wishes to change examiner before the next examination, a written application shall be sent to the department responsible for the course and shall be granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In cases where a course has been discontinued or has undergone major changes, the student shall normally be guaranteed at least three examination occasions (including the ordinary examination) during a period of at least one year from the last time the course was given.

Grades
The grading scale comprises: Pass with Distinction (VG), Pass (G) and Fail (U).
To pass the course the student needs to pass the oral presentation as well as the take-home exam.

To receive a higher grade than G/Pass the student needs to pass an individual oral examination, in addition to the oral presentation and the take-home exam. The final grade is decided from the combined results of the oral presentation, the take-home exam, and the oral exam.

Course evaluation
The course is evaluated through meeting after the course between teachers and student representatives. Further, an anonymous questionnaire is used to ensure written

information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

Additional information

Knowledge of Functional Programming is recommended, e.g., from the course DIT143 Functional Programming, or equivalent.

The course is a joint course together with Chalmers.

Course literature to be announced the latest 8 weeks prior to the start of the course.

The course replaces the course DIT232 *Types for Programs and Proofs*, 7.5 hec. The course cannot be included in a degree which contains DIT232. Neither can the course be included in a degree which is based on another degree in which the course DIT232 is included.