GHENT UNIVERSITY

Course Specifications

Valid as from the academic year 2024-2025

Reasoning, Abstracting and Formulating (C002908)

Course size *(nominal values; actual values may depend on programme)*

Credits 6.0 **Study time 180 h**

Course offerings and teaching methods in academic year 2024-2025

A (semester 1)	Dutch	Gent	lecture
			seminar

Lecturers in academic year 2024-2025

Laermans, Eric	TWO5	lecturer-in-charge

Offered in the following programmes in 2024-2025	**crdts**	**offering**
Bachelor of Arts in Moral Sciences	6	A
Bachelor of Arts in Philosophy	6	A
Bachelor of Science in Computer Science	6	A

Teaching languages

Dutch

Keywords

Elementary logic, sets and relations, induction, functional programming, reasoning about program code.

Position of the course

The main aim of this course is to bridge the gap between the skills that were learned in secondary school, essentially in the form of “mathematical thinking”, and those that are needed in the computer science curriculum. To this end, the course starts from a mathematics context, because students are already familiar with it, and because the three central components of the course can already be studied using the familiar mathematics. In the second part, the course moves on to a computer science context, taking reasoning about program code as the point of departure.

- The ability to reason correctly and fluently is of central importance for computer scientists. When developing software, for instance, a continuous alertness is needed to verify that a given specification is satisfied. Although such forms of reasoning often rely on human intuition, guarantees are sometimes required about the correctness of an implementation, in the form of an accurate and detailed proof. Moreover, being able to construct and recognize correct proofs is also a necessary skill for subsequent courses on mathematics and theoretical computer science.
- Abstracting refers to the ability to find solutions for concrete problems in which, to the extent possible, irrelevant details are not taken into account. The purpose of this process is to keep the complexity of difficult problems more manageable, and to find solutions that can be reused in different situations. Abstraction is paramount both for developing software, among others in the context of code reuse, and for theoretical computer science.
- Before any form of reasoning can take place, one has to correctly formulate and understand the formal specifications and definitions that are involved. For instance, one can only prove that a program satisfies a certain specification, but not that the formal specification is a correct translation of what is intuitively desired. Hence, the ability to know intuitively when a certain formulation is correct is also a fundamental skill for computer scientists.

Contents

- Elementary logic and truth tables: most important derivation rules from the proposition and predicate calculus; substitution, proof techniques.
- Sets and relations: terminology and notations; converting informal descriptions to formal specifications; converting formal specifications to informal descriptions.

- Induction and recursion: natural induction; structural induction; recursive definition of abstract data types.
- Functional programs: formal specification; reasoning about program code.

Initial competences

Mathematics from secondary school.

Final competences

1 Being able to explain the difference between an informal and formal definition/specification/proof, and understanding when an informal approach is sufficient.
2 Being able to recognize assumptions that are implicit or hidden in a problem formulation.
3 Being able to convert informal criteria to formal specifications, and vice versa. Being able to convert formal specifications to intuitive descriptions in natural language.
4 Being able to convert an intuitive idea for a proof to an actual proof, and vice versa. Being able to recognize the intuition that constitutes the basis of a given proof.
5 Being able to generalize specifications/definitions/proofs to a more abstract form.
6 Being able to construct simple proofs, also in abstract domains.
7 Being able to fluently apply the techniques that were introduced to verify the correctness of functional programs.

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Seminar, Lecture

Extra information on the teaching methods

The theory is taught during the lectures. The labs constitute pencil-and-paper exercises that allow the students to actively process the material with the help of a teaching assistant.

Study material

Type: Syllabus

Name: -
Indicative price: € 10
Optional: no
Language : Dutch
Number of Pages : 292
Available on Ufora : Yes
Additional information: Members of WiNa can purchase the course at a discount. The online version will be available via Ufora from mid-October.

Type: Slides

Name: -
Indicative price: Free or paid by faculty
Optional: no
Language : Dutch
Available on Ufora : Yes
Additional information: Available free of charge in electronic form

References

Richard Bird. Thinking Functionally with Haskell, CUP, 2014, ISBN: 978-1107452640.
Daniel J. Velleman. How To Prove It: A Structured Approach (second edition). Cambridge University Press, 2006, ISBN 978-0521675994
David Gries. The Science of Programming. Springer-Verlag New York, 1987, ISBN 978-0387964805

Course content-related study coaching

Students actively process the learning material while making exercises in the presence of a teaching assistant. Furthermore students can submit their solutions for additional exercises to get feedback from the assistant or the instructor. The assistant and the instructor are also available to students for additional individual explanation outside of the scheduled class times.

Assessment moments

end-of-term and continuous assessment

Examination methods in case of periodic assessment during the first examination period

Written assessment with open-ended questions

Examination methods in case of periodic assessment during the second examination period

Written assessment with open-ended questions

Examination methods in case of permanent assessment

Assignment

Possibilities of retake in case of permanent assessment

examination during the second examination period is not possible

Extra information on the examination methods

The non-periodical evaluation is based on a combination of one or more assigments for which the students submit solutions during the semester, and an intermediary test. These will be pencil-and-paper exercises.

Calculation of the examination mark

1st exam: periodical (80%) and permanent (20%) evaluation. 2nd exam: periodical evaluation (100%).
The score for permanent evaluation only counts if the student obtains at least 9/20 for his/her periodic evaluation.