# MAP-I
# Programa Doutoral em Informática

# *Program Semantics, Verification, and Construction*
Approaches to Correct Software

Unidade Curricular em Teoria e Fundamentos
*Theory and Foundations*
(UCTF)

DI-UM, DCC-FCUP

May 9, 2011

**Abstract**

This text presents a UCTF ("Unidade Curricular em Teorias e Fundamentos") course in the context of the joint PhD programme (Minho, Aveiro, Porto) in Informatics (MAP-I). The team responsible for the proposal consists of lecturers and professors from the Informatics department of the University of Minho (DI-UM) and the Computer Science Department - Faculty of Science of the University of Porto (DCC-FCUP).

**LECTURING TEAM**

| | |
|---|---|
| **DIUM:** | Maria João Frade, José Nuno Oliveira |
| **DCCFCUP:** | Sandra Alves, Sabine Broda |

# 1 Course Description

## 1.1 Subject and Context

The reliability of computing systems plays an essential role in modern society, where so many areas of human activity depend on technology. The deliverables of software projects may no longer be limited to code; the ability to produce *certified code* is now crucial. Code may be certified as being *functionally correct*, or as possessing certain execution properties (for instance, a program may be certified as not trying to access unauthorised resources).

The ability to certify software in this way requires a sound knowledge of the theory of programming languages and mathematical reasoning tools, as well as acquaintance with tool-assisted techniques. Once the requirements have been identified, the challenge is to be able to formalise and prove the corresponding properties (functional, security, or safety), choosing from a number of different conceptual tools and different notions of certificate.

This course gives an overview of the theory of programming languages at an advanced level (which will help to cancel the heterogeneous backgrounds of students on these subjects) and then goes on to apply the theory to methods for obtaining correct, certified software.

The second part of the course, on Program Verification, contains material analogous to what is taught in many standard advanced-level courses on program verification, with the additional coverage of recent results and current research. The third part covers the "correct by construction" approach, such as followed in courses taught at the universities of Oxford or Nottingham. In this approach certification is an automatic byproduct of the code development process.

**ACM Computing Classification System subjects covered:**

- /Theory of Computation/MATHEMATICAL LOGIC AND FORMAL LANGUAGES/Mathematical Logic/
- /Theory of Computation/COMPUTATION BY ABSTRACT DEVICES/
- /Theory of Computation/LOGICS AND MEANINGS OF PROGRAMS/Semantics of Programming Languages/
- /Theory of Computation/LOGICS AND MEANINGS OF PROGRAMS/Specifying and Verifying and Reasoning about Programs/
- /Software/SOFTWARE ENGINEERING/Software/Program Verification/

## 1.2 Objectives

This UCTF aims

- to present in a systematic way a vast set of results in fundamental areas of Theoretical Computer Science, in particular Logic, $\lambda$-calculus, Type Theory, and Programming Language Semantics, as well as the relationships between them;
- to achieve learning outcomes in the rigorous approaches to the production of correct software, namely
  - in *Program Verification*, the activity that aims to establish that a program effectively behaves according to its specification, or that its behaviour is characterized by a set of given properties;
  - in *Mathematical Program Construction*, a method for obtaining correct programs from specifications, strongly based on *Program Calculation*.

## 1.3 Learning Outcomes

- To understand the relation between Intuitionistic Logic and Type Theory.
- To use languages with simple, dependent, polymorphic, or inductive types, for programming, expressing properties, or writing specifications.
- To understand the use of the operational, denotational, and axiomatic styles of semantics in different contexts.
- To use proof assistants for conducting formal proofs interactively.
- To express and prove properties of functional and imperative programs with the help of proof assistants and verification condition generators.
- To understand current trends in Program Verification techniques and approaches to the certification of program properties.
- To understand the dichotomy between specification and implementation in software design.
- To understand that software (implementations) can be calculated by solving systems of equations (specifications) as in other branches of science and engineering.
- To appreciate the calculational power of the PF-transform and of the underlying allegory of binary relations.

## 1.4 Syllabus

- Chapter I: Overview of Foundations (18 hours)
    1. Propositional and First-order logic
    2. Intuitionistic logic
    3. Natural deduction
    4. First-order theories
    5. $\lambda$-calculus (terms, reduction, the Church-Rosser Theorem)
    6. Simple Types (Church versus Curry typing, normalization, extensions)
    7. The Curry-Howard isomorphism
    8. Introduction to operational semantics
    9. Domain theory (complete partial orders, continuous functions)
    10. Denotational semantics
- Chapter II: Program Verification (18 hours)
    1. Dependent Types:
        - First-order dependent types
        - Type equivalence
        - Sum types
        - The calculus of inductive constructions
        - Programming with dependent types
    2. Type-based proof assistants
        - Interactive proof development
        - Tactics and tacticals
        - Inductive data types and predicates
    3. Program correctness: specification; partial and total correctness
    4. Verification of the correctness of functional programs:
        - Extraction of the computational contents of a correctness proof
        - Using programs for structuring correction proofs
    5. Axiomatic semantics of imperative programs:
        - Assertions; semantics of assertions
        - Hoare proof rules for correctness
    6. Tool support for the specification, verification, and certification of programs:
        - Proof assistants

    - Verification condition generators

7. Survey of alternative approaches to program verification:
    - Abstract machine-based approaches
    - Certifying compilation and proof-carrying code

- Chapter III: Program Construction (12 hours)

1. Introduction to the mathematics of program construction
    - The specification / implementation dichotomy. Abstract modeling.
    - Correct by verification versus correct by construction.
2. Description versus calculation
3. The Point-free (PF) transform
    - Taxonomy of binary relations; simple relations and their role in abstract modeling
    - 'Point-free' notation and reasoning
    - Rules of the PF-transform
    - Categorical and allegorical foundations
4. Universal properties and Galois connections
    - Universal constructions and properties; natural properties
    - Reynolds' relation and the free-theorem of polymorphism
    - Galois connections and their corollaries
5. Reasoning by PF-calculation
    - PF-calculation of the consistency of a formal model: satisfiability and invariance
    - Data-level calculation: representing and abstracting data models.
6. Inductive program calculation
    - Relational hylomorphisms
    - Fixpoint calculus and Galois connections: the fixpoint fusion theorem
    - Calculating recursive solutions for hylo-equations
7. Open issues and hot topics in the mathematics of program construction

## 1.5 Teaching Methods

- Lectures
- Occasional tool demonstration / case study sessions

## 1.6 Student Assessment

- Examinations
- Research assignments, which may include a talk given on a suggested paper, or practical assignments

## 2 Lecturing Team

The team consists of members of the Department of Informatics of the University of Minho and the Department of Computer Science of the University of Porto (Faculty of Science).

All team members are working, and have worked actively in the past few years, on topics that are directly related to the subjects covered by this course, as detailed below.

- Sandra Alves has worked on $\lambda$-calculus and type theory. Her work has been focusing on several aspects of linearity in programming languages, in particular, the use of linearisation in the definition of efficient simulations of functional programming languages.
- Sabine Broda (DCC-FCUP) has worked on Mathematical Logic, $\lambda$-calculus, and Type Theory.
- Maria J. Frade (DI-UM) has worked on $\lambda$-calculus, type systems, and Proof Theory.
- José N. Oliveira (DI-UM) has worked extensively on Formal Methods in Software Engineering and is in fact a pioneer of this area in Portugal. Recently he has become interested in the calculation-based approach to program construction.