

# Formal Systems and their Applications

Course nr: B-KUL-H04H8B

Frank Piessens  
([Frank.Piessens@cs.kuleuven.be](mailto:Frank.Piessens@cs.kuleuven.be))

# Course Overview

# Formal Systems

- A Formal System is a mathematically precise system that *models* some phenomenon of interest
  - A formal language, and
  - Some rules or axioms to “reason about” or “compute with” the system
- Formal systems permeate mathematics and computer science (and other exact sciences):
  - The natural numbers (or the integers, rational numbers, ...)
  - First-order logic (and other logics)
  - Programming languages
  - Modeling languages
  - ...

# Formal Systems

- A “good” formal system:
  - Should be “simple”: so we can reason about it and prove properties
  - Should be “useful”: it should model sufficient detail about the phenomenon under consideration such that what we learn from studying the system applies to the “real thing”

# Formal Systems in CS

- A plethora of computer science topics have been studied scientifically using formal models:
  - Programming languages
  - Type systems
  - Concurrency
  - Cryptography
  - Security protocols
  - Access control
  - Knowledge representation
  - ...

# This course

- In this course, we will study formal models for programming languages and type checkers
  - The programming language is the primary tool of a computer scientist
  - Type checking is one of the most successful applications of formal methods in computer science
    - Detect errors early
    - Enforce abstractions
    - Improve code readability
    - Guarantee safety
    - Improve efficiency

# Course Overview

- Introduction
- Formal models of programming languages
  - Abstract syntax trees
  - Structural operational semantics
  - The lambda calculus
- Simple type systems
  - Typing an expression language
  - Simply typed lambda calculus
  - Typing simple programming language features: records, variants, lists, references, exceptions, ...

# Course Overview

- Subtyping
    - Structural subtyping
    - Interaction of subtyping with other language features
    - Typing object oriented language features
    - Nominal subtyping
  - Conclusion
- 
- Course project:
    - implementation of an interpreter and type checker in the Scala programming language
    - Gain some experience with a dependently typed language

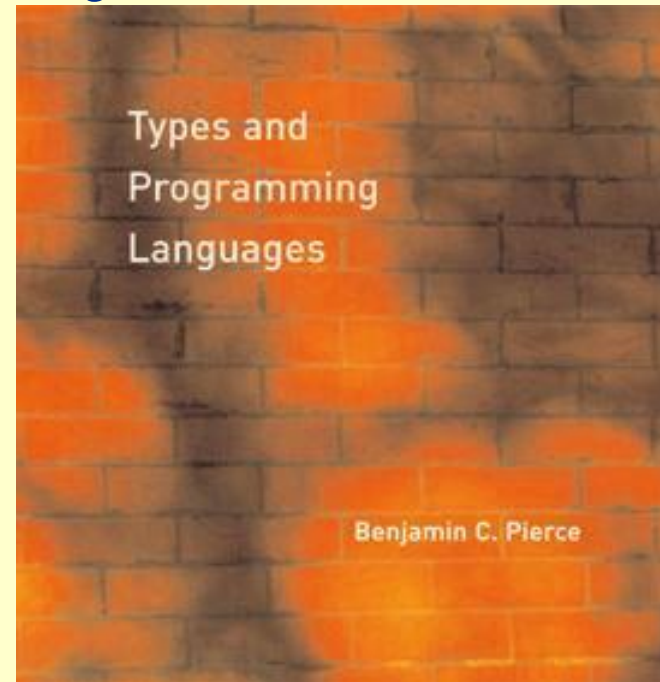
# Administrative Staff

# Lectures and project

- 10-12 lectures
  - Theory and simple exercises
- Project
  - 2-3 guided sessions on programming in Scala / Agda and on the project assignment
  - 30 hours of (team-)work on solving the project assignment
- Examination
  - Classic “closed-book” examination on the theory
  - Defense of your project

# Study material

- Course book:
  - *Types and Programming Languages* – Benjamin Pierce
- Slides
- Scala reference materials
- Agda reference materials
- Course website
  - <http://www.cs.kuleuven.be/~frank/FST/>



Any Questions?