HOGESCHOOL ROTTERDAM / CMI

Advanced Programming

INFSEN02-8/INFSEN23-8
2018-2019

Number of study points:	4 ects
Course owners:	Francesco Di Giacomo, Giuseppe Maggiore, Mohamed Abbadi

Module description

Module name:	Advanced Programming
Module code:	INFSEN02-8/INFSEN23-8
Study points and hours of effort:	This module gives 4 ects, in correspondence with 112 hours: • 2 X 7 hours of combined lecture and practical • the rest is self-study
Examination:	Written Exam and Practical assessment
Course structure:	Lectures, self-study, and practical exercises
Prerequisite knowledge:	all INFDEV courses.
Learning materials:	• Book: Don Syme - Expert F# • exercises and assignments, to be done at home and during the practical part of the lectures (pdf): found on N@tschool
Connected to competences:	realiseren en ontwerpen
Learning objectives:	At the end of the course, the student: • **understands** the fundamental semantic difference between functional and imperative programming. `(FP VS IMP)` • **understands** reduction strategies such as $\rightarrow_\beta$. `(RED)` • **understands** the basics of a functional type system. `(TYP)` • **can program** with the typical constructs of a modern functional language. The language of focus is F#. `(FP EXT)`
Course owners:	Francesco Di Giacomo, Giuseppe Maggiore, Mohamed Abbadi
Date:	May 7, 2019

1 General description

Functional programming and functional programming languages are increasing in popularity for multiple reasons and in multiple ways, to the point that even mainstream languages such as Python, C++, C#, and Java are being extended with more and more functional programming features such as tuples, lambda's, higher order functions, and even monads such as LINQ and async/await. Whole architectures such as the popular map/reduce are strongly inspired by functional programming.
"Java™ developers should learn functional paradigms now, even if they have no immediate plans to move to a functional language such as Scala or Clojure. Over time, all mainstream languages will become more functional" [IBM].
"LISP is worth learning for a different reason — the profound enlightenment experience you will have when you finally get it. That experience will make you a better programmer for the rest of your days, even if you never actually use LISP itself a lot." – Eric S. Raymond
"SQL, Lisp, and Haskell are the only programming languages that I've seen where one spends more time thinking than typing." – Philip Greenspun
"I do not know if learning Haskell will get you a job. I know it will make you a better software developer." – Larry O' Brien
The reason for this growth is to be found in the safe and deep expressive power of functional languages, which are capable of recombining simpler elements into powerful, complex other elements with less space for mistakes and more control in the hands of the programmer. This comes at a fundamental cost: functional languages are structurally different from imperative and object oriented languages, and thus a new mindset is required of the programmer that wishes to enter this new world. Moreover, functional languages often require more thought and planning, and are thus experienced, especially by beginners, as somewhat less flexible and supporting of experimentation.

1.1 Relationship with other didactic units and required knowledge

This module completes and perfects the understanding and knowledge of programming that was set up in the Development courses of the first year. The preliminary knowledge necessary to fully understand this course covers the following topics:

- Semantics of programming languages and its evaluation.
- The memory model of *Stack* and *Heap*.
- *Type systems* and type checking.
- Dynamic vs Static typing.

2 Course program

The course is structured into eight lectures. The eight lectures take place during the eight weeks of the course, but are not necessarily in a one-to-one correspondance with the course weeks.

Unit 1

Topics

- Stateful vs stateless computation
- Lambda calculus semantics.
  - Variables
  - Lambda-abstractions/functions
  - Function application
- Introduction to F#

Unit 2

Topics

- Typed lambda calculus. Typing variables, lambda abstractions, function applications.
- Basic data structures in F#
- Pattern matching.

Unit 3

Topics

- Polymorhpism in functional programming
- Discriminated unions.

Unit 4

Topics

- Higher-order functions.
- Function composition.
- HOF design patterns.

Unit 5

Topics

- Hierarchical advanced data structures.
- Immutable trees.
- Immutable binary search trees.
- Decision trees.

3 Assessment

The course is tested with two exams: a written exam and a practical exam. Both parts happen exclusively on paper and must be sufficient in order to pass the course.

3.1 Theoretical examination INFSEN02-8/INFSEN23-8

The general shape of an exam for `INFSEN02-8/INFSEN23-8` is made up of a short series of highly structured open questions. In each exam the content of the questions will change, but the structure of the questions will remain the same. Questions might include (but not limited to): apply the semantics of lambda calculus on a small function, determine the type of a functional program, determine the result of the execution of a functional program. A sample exam will be provided during the course.

3.2 Practical examination INFSEN02-8/INFSEN23-8

The practical exam requires to implement a series of functions in the language F# described in the exam text.

Appendix 1: Assessment matrix

Learning objective	Dublin descriptors
`FP VS IMP`	1, 4, 5
`RED`	1, 2, 4, 5
`TYP`	1, 4, 5
FP EXT	1, 2

Dublin-descriptors:

1. Knowledge and understanding
2. Applying knowledge and understanding
3. Making judgments
4. Communication
5. Learning skills