Functional Programming

Anders Kalhauge and Jens Egholm Petersen

Spring 2018

The goal of this course is to introduce the students to functional programming. Programming functionally requires another mind-set than procedural and especially object-oriented programming. The students will use functional programming in practice, and will be trained to decide where to use which paradigm.

After living in the dark for several decades, functional programming is having a revival. The demand for parallelism, due to the change in CPU architecture with more and more cores, are challenging languages as C# and Java. Functional programming handles parallel programming very elegantly and efficiently. The newest versions of object oriented languages introduce functional programming constructs to support parallelism. JavaScript is a good example of such a language, that is spreading to more and more platforms. Knowledge of functional programming will be an increasing demand in the future.

Course content

The course will look at functional programming and will introduce the languages Lisp, Elm and Haskell to the students. The course will start out by looking at recursion, CPU architecture and lists, followed by a dive into client programming with Elm and server programming with Haskell. As a part of the Haskell lectures, concurrency and parallelism will be covered and the students will in the end of the course write an application (client and server) entirely in functional languages.

The following theoretical topics are covered:

- Recursion and tail recursion
- Lambda calculus
- Higher-order functions
- Currying and partial application
- CPU architecture and call stacks
- Linked lists
- Data modelling
- Concurrency and parallelism
- Introduction to category theory

0.1 Learning goals

At the end of the course the students will:

- Have a general knowledge about the functional paradigm
- Knows the building blocks of a functional programming language
- Knows how to support parallelism using a functional language
- Knows the overall constructs in a functional language
- Knows where to find additional information

At the end of the course the students can:

- Write basic web applications using Elm
- Understand and write programs written in LISP
- Understand and write simple programs written in Haskell

The students will in groups make four major assignments in central topics of the curriculum. The solution of one of these will form the basis for the exam. Also the students will do eight smaller assignments.

Exam

The exam is oral. The student will prepare a (app. ten minutes) presentation of the solution of one of the major assignments. Further discussions will be based on the presentation, but can include all aspects of the curriculum.

Admission requirements

In order to be approved for the exam:

- All four major assignments must be handed in
- At least 80 study points must be obtained

Study points

- Hand in of major assignments (15 per assignment): 60
- Hand in of minor assignments (10 per assignment): 40