COMP4075: Lecture 1

Administrative Details and Introduction

Henrik Nilsson

University of Nottingham, UK

Finding People and Information (1)

- Henrik Nilsson
  Room A08, Computer Science Building
  e-mail: `nhn@cs.nott.ac.uk`

Finding People and Information (1)

- Henrik Nilsson
  Room A08, Computer Science Building
  e-mail: `nhn@cs.nott.ac.uk`
- Main module web page:
  `www.cs.nott.ac.uk/~psznhn/COMP4075`
  (or `www.cs.nott.ac.uk/~psznhn/G54RFP`)

Finding People and Information (1)

- Henrik Nilsson
  Room A08, Computer Science Building
  e-mail: `nhn@cs.nott.ac.uk`
- Main module web page:
  `www.cs.nott.ac.uk/~psznhn/COMP4075`
  (or `www.cs.nott.ac.uk/~psznhn/G54RFP`)
- Moodle (COMP4075):
  `moodle.nottingham.ac.uk/course/view.php?id=106278`
- Microsoft Teams: COMP4075-RFP
  (code 951kkp to join)

Finding People and Information (2)

Questions via:

- Microsoft Teams
- Moodle COMP4075 forum

Finding People and Information (2)

Questions via:

- Microsoft Teams
- Moodle COMP4075 forum

General discussion is encouraged on the open Teams channel and the forum. But for obvious reasons, while anything related to the module including the coursework can be discussed, exact solutions to the coursework must not be posted.

Module Activities (1)

- Cheifly on-line delivery:
  - Pre-recorded lectures
  - Slides, notes, etc. available for download
  - Weekly live Q&A session via Teams Fridays 11:00–12:00

Module Activities (1)

- Cheifly on-line delivery:
  - Pre-recorded lectures
  - Slides, notes, etc. available for download
  - Weekly live Q&A session via Teams Fridays 11:00–12:00
- Occsionally live lectures:
  - Possibly guest lecture, because might work better for specific topic, or just for varition
  - The Friday 11:00–12:00 slot unless otherwise indicated, with Q&A afterwards

Module Activities (2)

- It may be feasible to occasionally have some face-to-face interaction:
  - Computing Fridays 11:00–13:00 is nominally scheduled in B52, CS building
  - It might be possible to run mixed face-to-face/virtual Q&A from there

Aims and Motivation (1)

- To introduces tools, techniques, and theory needed for programming real-world applications functionally.

Aims and Motivation (1)

- To introduces tools, techniques, and theory needed for programming real-world applications functionally.
- Particular emphasis on the inherent benefits of functional programming and strong typing for:
  - reuse
  - maintenance
  - concurrency
  - distribution
  - scalability
  - high availability

Aims and Motivation (2)

- Such aspects have:
  - contributed to the popularity of functional programming for demanding applications e.g. in the finance industry
  - have had a significant impact on the design of many languages and frameworks such as Java, C#, and Rust, MapReduce, React

Aims and Motivation (3)

We will use Haskell as medium of instruction, but:

- What is covered has broad applicability
- Fundamentally, this is *not* a module teaching advanced Haskell as such, even if there inevitably will be some aspects of that
- Guest lectures (hopefully) and optional project provide opportunities to branch out beyond Haskell

Who should take this module

- Anyone who wants to learn about (some of) the kind of functional programming techniques that are relevant for solving real-world programming problems.

Who should take this module

- Anyone who wants to learn about (some of) the kind of functional programming techniques that are relevant for solving real-world programming problems.
- Anyone who wants to get a better understanding of applying a functional approach to real-world programming problems often is a good idea.

Who should take this module

- Anyone who wants to learn about (some of) the kind of functional programming techniques that are relevant for solving real-world programming problems.
- Anyone who wants to get a better understanding of applying a functional approach to real-world programming problems often is a good idea.
- Other reasons? Why did *you* opt to take this module?

Prerequisites

- The modules
  - COMP1009 Programming Paradigms
  - COMP2003 Advanced Functional Programming

  or equivalent assumed

Prerequisites

- The modules
  - COMP1009 Programming Paradigms
  - COMP2003 Advanced Functional Programming

  or equivalent assumed
- Feasible also for someone with general strong programming background as long as:
  - Solid experience of more than one programming paradigm
  - Prepared and realistiaclly able to put in the required extra time and effort

Content (1)

The module will cover a range of topics, some more foundational, some applied, such as:

- Lazy functional programming
- Purely functional data structures
- Key libraries
- Functional design patterns
- Concurrency
- Web programming
- GUIs

Content (2)

However:

Content (2)

However:

- Impossible to systematically cover ***everything*** you need to know for real-world functional programming

Content (2)

However:

- Impossible to systematically cover ***everything*** you need to know for real-world functional programming
- As a result, the module covers a mix of somewhat distinct topics

Content (2)

However:

- Impossible to systematically cover ***everything*** you need to know for real-world functional programming
- As a result, the module covers a mix of somewhat distinct topics
- But, drawing from my own industrial experience working 80% as a developer in a bank writing financial software functionally, I can guarantee that what is covered is genuinely relevant and useful.

Guest Lectures?

- Trying to arrange a couple of guest lectures
- However, on-line delivery does raise extra concerns for many industrial pratitioners, so we will have to see what is possible

Literature (1)

No main reference. The following two will be useful, though, both freely available online:

- *Haskell*, Wikibooks
- *Real World Haskell*, by Bryan O'Sullivan, John Goerzen, and Don Stewart

We will also use tutorials, research papers, videos, etc. References given on module web page as we go along.

Assessment

- 100 % coursework

Assessment

- 100 % coursework
- Coursework, 2 parts:
  - Part I: Basics; 15 h
  - Part II: Advanced topics and applications; 35 h

Assessment

- 100 % coursework
- Coursework, 2 parts:
  - Part I: Basics; 15 h
  - Part II: Advanced topics and applications; 35 h
- Part I will be released in a week or two

Assessment

- 100 % coursework
- Coursework, 2 parts:
  - Part I: Basics; 15 h
  - Part II: Advanced topics and applications; 35 h
- Part I will be released in a week or two
- Use time until then to get up to speed on Haskell.

Assessment

- 100 % coursework
- Coursework, 2 parts:
    - Part I: Basics; 15 h
    - Part II: Advanced topics and applications; 35 h
- Part I will be released in a week or two
- Use time until then to get up to speed on Haskell.
- Further details TBA

Assessment

- 100 % coursework
- Coursework, 2 parts:
  - Part I: Basics; 15 h
  - Part II: Advanced topics and applications; 35 h
- Part I will be released in a week or two
- Use time until then to get up to speed on Haskell.
- Further details TBA

Start early! It is ***not*** possible to do this coursework at the last minute.

COMP4095: Optional Project (1)

- 10 credits (100 hours)

COMP4095: Optional Project (1)

- 10 credits (100 hours)
- Opportunity to learn in depth about aspects of functional programming at scale.

COMP4095: Optional Project (1)

- 10 credits (100 hours)
- Opportunity to learn in depth about aspects of functional programming at scale.
- Project must be clearly related to what is covered in COMP4075, but “functional” interpreted in a broad sense.

COMP4095: Optional Project (1)

- 10 credits (100 hours)
- Opportunity to learn in depth about aspects of functional programming at scale.
- Project must be clearly related to what is covered in COMP4075, but “functional” interpreted in a broad sense.
- Project defined through a “pitch” that must be discussed and agreed. Needs to clarify:
  - The relevance of the project to COMP4075
  - Size appropriate for 10 credits

COMP4095: Optional Project (2)

Preliminary timeline (TBC):

- Release of project criteria: late October
- Pitch deadline: late November (but earlier is better)
- Submission deadline (code and report): mid January