Advanced Programming

Course Presentation

Troels Henriksen

2024

Why I think this course is worthwhile

The purpose of this course is to provide practical experience with sophisticated programming techniques and paradigms from a language-based perspective. The focus is on high-level programming and systematic construction of well-behaved programs.

- from course description

- Solving complex problems in a **maintainable** and **reusable** way.
- Allowing principled **reasoning** about program behaviour.
- Performing rigorous **testing**.

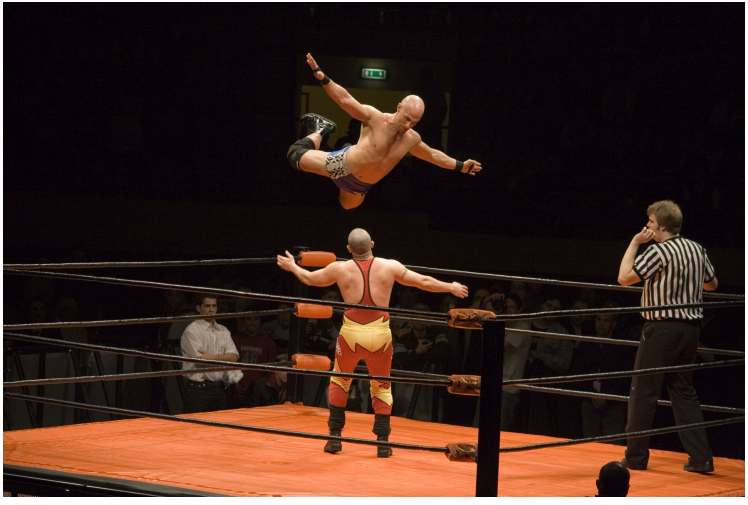
Why I think this course is worthwhile

The purpose of this course is to provide practical experience with sophisticated programming techniques and paradigms from a language-based perspective. The focus is on high-level programming and systematic construction of well-behaved programs.

- from course description

- Solving complex problems in a **maintainable** and **reusable** way.
- Allowing principled **reasoning** about program behaviour.
- Performing rigorous **testing**.
- Pushing **separation of concerns** further than you have seen before.
- **This is how I program!**

Advanced Programming and Professional Wrestling



Christopher Daniels performing a *flying crossbody* on Jonny Storm.

Photo by Hugo Fernandes

https://commons.wikimedia.org/wiki/File:Christopher_Daniels_1.jpg

Kayfabe

The fact or convention of presenting staged performances as genuine or authentic.

Kayfabe

The fact or convention of presenting staged performances as genuine or authentic.

The Kayfabe of AP

- Principled design works better the larger your system is and the longer it has to be maintained.
- For practical reasons we look at small systems that are not maintained over time and that would probably work if implemented any reasonable way.
- We pretend the systems we study are a lot bigger than they actually are.

Which languages and tools you will use

- All programming is in **Haskell**.
  - Purely functional.
  - Somewhat similar to F#.
  - Lazy.
- We use standard compilers and build tools.
  - Feel free to use fancy editor integration, but it is not required.
  - Tool expertise not part of learning goals; we try to streamline as much as possible.
  - Windows users will (probably) have to use WSL2.
- We assume programming proficiency, roughly corresponding to an undergraduate degree in computer science.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.
5. How to structure (potentially large) concurrent systems in a robust manner.

How we hope this course will change your life

Alternatively: *why is this course interesting even your main interest is AI, algorithmics, user interfaces, or bossing people around?*

1. Reasoning about computation as a mathematical construct.
2. Clean separation and modularisation.
3. Taking generalisation and abstraction further.
4. How to parse input data in a principled and convenient manner.
5. How to structure (potentially large) concurrent systems in a robust manner.
6. How to effectively test complex systems without writing a million unit tests.

The Course Team

Lecturers

-  Troels: **course responsible**, Haskell, parsing, free monads, concurrency.
-  Mikkel: type classes, monads, property-based testing.

TAs

Robert 

Francisco 

Therese 

Joachim 

Mikkel

Ian 

Rasmus

Thomas 

The didactic architecture of AP

- Each week has **lectures**, **course notes** and **other reading material**…
- …which enable you to solve programming **exercises**…
- …the solutions of which form the basis for the **assignments**.

The idealised AP study algorithm

The idealised AP study algorithm

1. Repeat for every week:

The idealised AP study algorithm

1. Repeat for every week:
   1.1 Consume learning material:
   - Attend lectures.
   - Read texts.

The idealised AP study algorithm

1. Repeat for every week:
   - 1.1 Consume learning material:
     - Attend lectures.
     - Read texts.
   - 1.2 Solve exercises.

The idealised AP study algorithm

1. Repeat for every week:
   - 1.1 Consume learning material:
     - Attend lectures.
     - Read texts.
   - 1.2 Solve exercises.
   - 1.3 Solve group assignment.

The idealised AP study algorithm

1. Repeat for every week:
   - 1.1 Consume learning material:
     - Attend lectures.
     - Read texts.
   - 1.2 Solve exercises.
   - 1.3 Solve group assignment.
   - 1.4 Hand in assignment.

The idealised AP study algorithm

1. Repeat for every week:
   - 1.1 Consume learning material:
     - Attend lectures.
     - Read texts.
   - 1.2 Solve exercises.
   - 1.3 Solve group assignment.
   - 1.4 Hand in assignment.
2. Attend exam.

The idealised AP study algorithm

1. Repeat for every week:
   1.1 Consume learning material:
      - Attend lectures.
      - Read texts.
   1.2 Solve exercises.
   1.3 Solve group assignment.
   1.4 Hand in assignment.
2. Attend exam.
3. **Optional**: Attend re-exam.

The idealised AP study algorithm

1. Repeat for every week:
   - 1.1 Consume learning material:
     - Attend lectures.
     - Read texts.
   - 1.2 Solve exercises.
   - 1.3 Solve group assignment.
   - 1.4 Hand in assignment.
2. Attend exam.
3. **Optional**: Attend re-exam.
4. **Optional**: Return to 1.

The idealised AP study algorithm

1. Repeat for every week:
   1.1 Consume learning material:
      - Attend lectures.
      - Read texts.
   1.2 Solve exercises.
   1.3 Solve group assignment.
   1.4 Hand in assignment.
2. Attend exam.
3. **Optional**: Attend re-exam.
4. **Optional**: Return to 1.

Weekly resource allocation

- **Lectures:** 4 hours
- **Reading:** 4 hours
- **Exercises:** 4 hours
- **Assignment:** 8 hours

Individual course elements

- **Exercises**, weekly, voluntary but *strongly recommended*.
- **Assignments**:
  - Six in total.
  - Solved in groups of up to 3 students.
  - Awarded 0-4 points.
  - **No resubmissions.**
- **Exam**: take-home, similar to assignments but larger. To qualify, you must have:
  - At least twelve points, and
  - at least one point in each assignment.
- **Lectures**, Tuesday and Thursday.
- **Exercise classes**, Thursday.
- **Study café**, Friday.
- **Course website**: `https://github.com/diku-dk/ap-e2024-pub`
- **Absalon**: used for handins, announcements, and discussions.
- **Discord**: `https://discord.gg/dJgTJ7mry7`

Schedule

When

	Monday	Tuesday	Wednesday	Thursday	Friday
10-12		Lecture		Exercises	
13-15				Lecture	Café
15-17				Exercises	

Where

Tuesday lecture: Aud 01, HCØ

Thursday lecture: Store UP1, DIKU

Friday café: Lille UP1, DIKU

Exercise classes: All over, see course webpage.

Learning material

No mandated textbook.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don't like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don't like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.
- We will provide some classic *research papers*.
  - You can make do without reading these, but they often provide very good background material to help your understanding.

Learning material

No mandated textbook.

But…

- We recommended chapters from *Programming in Haskell* by Graham Hutton.
  - If you don't like its style, feel free to read equivalent chapters from books such as *Learn You a Haskell for Great Good* or *Real World Haskell*, or any other that the Internet recommends.
- We will provide some classic *research papers*.
  - You can make do without reading these, but they often provide very good background material to help your understanding.
- We have written *course notes*.
  - `https://github.com/diku-dk/ap-notes`
  - One chapter per week; the idea is that they roughly correspond to lecture content.
    - And things that don't fit anywhere else.
  - Newly written this year, still WIP, please report bugs if you find any.
    - Incomprehensible text is a bug.

Preparing for the course

- Install necessary software.
  - `https://github.com/diku-dk/ap-e2024-pub/blob/main/haskell.md`
- Become a proficient functional programmer.
  - Do basic Haskell exercises, such as the ones from *Programming in Haskell* or *Learn You a Haskell for Great Good* (free online).
  - Or get a head start on the weekly exercises: `https://github.com/diku-dk/ap-e2024-pub/tree/main/week1`

Despite our emphasis on principles, this is a programming-intensive course where you are ultimately evaluated on your ability to produce working programs.

Questions?