Workshop on Trends in Functional Programming in Education

Flipped Graduate Classroom in a Haskell-based Software Testing Course

Onderwijsgek, Empty classroom, 2011. CC-BY-SA.

Jan van Eijck and Vadim Zaytsev

Master Software Engineering

- One year Master of Science programme at UvA
- Drifted away from computer science
- Courses taught:
  - software construction, evolution, testing
  - architecture, process, requirements
- Programmer in, software engineer out

http://www.software-engineering-amsterdam.nl

Software [Specification &] Testing

- The FM view on software engineering
  - ideas $\Longleftrightarrow$ models/specs $\Longleftrightarrow$ programs
  - logic reasoning, math notation, …
- The FP view on software
  - focus on the data flow instead of boilerplate
- Type systems
  - as an example of a software system specification

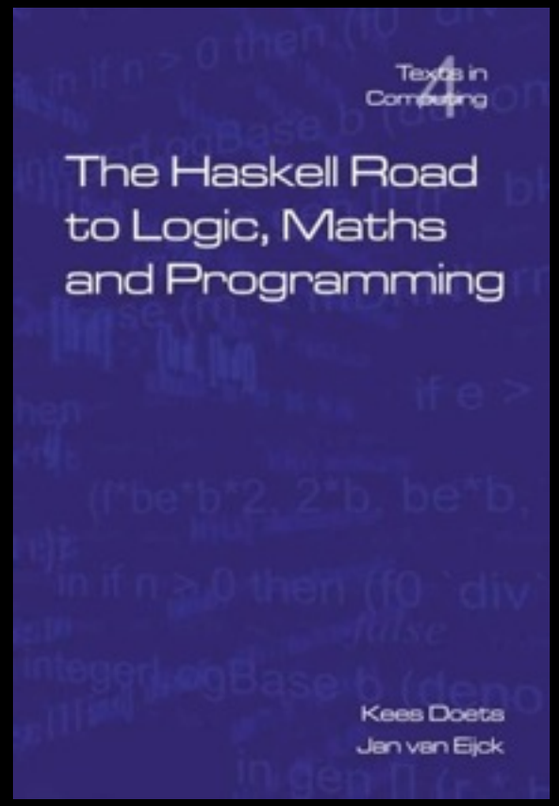
Learning Objectives [Bloom]

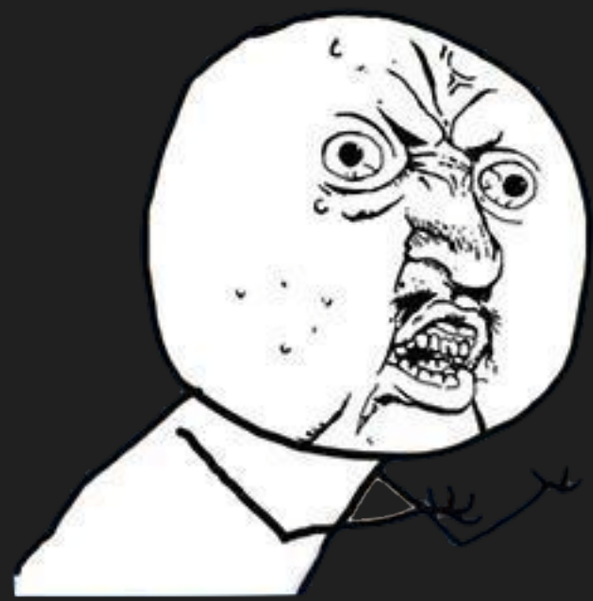
- (1) to recognise various testing techniques applied in SE
- (2) to compare testing techniques by applicability in certain scenarios
- (3) to implement formal specifications of software systems and test such systems for conformance
- (4) to differentiate among alternative approaches to testing
- (5) to argue in favour of one testing approach against another
- (5) to judge efficiency of a given model for testing

Anderson, Krathwohl, Airasian, Cruikshank, Mayer, Pintrich, Raths, Wittrock, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, 2000.

Current form (2013)

- two months, two full days a week (6 EC)
- a book;
- a weekly lecture (one group of 60 students);
- a weekly workshop (two hours, in groups of 20 students);
- 1½ day programming practical studies [obligatory]
- weekly sets of assignments (Haskell, in groups of 2–5)
- relatively positive evaluation by students





WHY THE CHANGE?

Example

Question 3 Let R be a binary relation on A. Two elements x and y of A are called *weakly R-connected* if there is a path of forward or backward R steps from x to y. It is allowed that this path is empty, so every point is weakly R connected to itself.

Suppose a function `tc :: Ord a => Rel a -> Rel a` for the transitive closure of a relation and a function `inv :: Ord a => Rel a -> Rel a` for inverting a relation are given. Use these to define a function

```
wConnected :: Ord a => Rel a -> a -> a -> Bool
wConnected r x y = ...
```

- $x == y$ || *elem (x,y) (tc (r ++ (inv r)))*
- $x = y$ v $(x,y) \in (R \cup R^{-1})^{+}$

Jan van Eijck, *Paper Exam Software Specification and Testing*, 21 October 2013.

Example

Question 3 Let R be a binary relation on A. Two elements x and y of A are called *weakly R-connected* if there is a path of forward or backward R steps from x to y. It is allowed that this path is empty, so every point is weakly R connected to itself.

Suppose a function `tc :: Ord a => Rel a -> Rel a` for the transitive closure of a relation and a function `inv :: Ord a => Rel a -> Rel a` for inverting a relation are given. Use these to define a function

```
wConnected :: Ord a => Rel a -> a -> a -> Bool
wConnected r x y = ...
```

- $x == y$ || *elem (x,y) (tc (r ++ (inv r)))*
- $x = y$ v $(x,y) \in (R \cup R^{-1})^{+}$

Jan van Eijck, *Paper Exam Software Specification and Testing*, 21 October 2013.

Example

Question 3 Let R be a binary relation on A. Two elements x and y of A are called *weakly R-connected* if there is a path of forward or backward R steps from x to y. It is allowed that this path is empty, so every point is weakly R connected to itself.

Suppose a function `tc :: Ord a => Rel a -> Rel a` for the transitive closure of a relation and a function `inv :: Ord a => Rel a -> Rel a` for inverting a relation are given. Use these to define a function

```
wConnected :: Ord a => Rel a -> a -> a -> Bool
wConnected r x y = ...
```

- $x == y$ || $elem\ (x,y)\ (tc\ (r\ ++\ (inv\ r)))$
- $x = y$ v $(x,y) \in (R \cup R^{-1})^{+}$

Jan van Eijck, *Paper Exam Software Specification and Testing*, 21 October 2013.

Example

Question 3 Let R be a binary relation on A. Two elements x and y of A are called *weakly R-connected* if there is a path of forward or backward R steps from x to y. It is allowed that this path is empty, so every point is weakly R connected to itself.

Suppose a function `tc :: Ord a => Rel a -> Rel a` for the transitive closure of a relation and a function `inv :: Ord a => Rel a -> Rel a` for inverting a relation are given. Use these to define a function

```
wConnected :: Ord a => Rel a -> a -> a -> Bool
wConnected r x y = ...
```

- $x == y$ || $elem\ (x,y)\ (tc\ (r\ ++\ (inv\ r)))$
- $x = y$ v $(x,y) \in (R \cup R^{-1})^{+}$

Jan van Eijck, *Paper Exam Software Specification and Testing*, 21 October 2013.

Example

Question 3 Let R be a binary relation on A. Two elements x and y of A are called *weakly R-connected* if there is a path of forward or backward R steps from x to y. It is allowed that this path is empty, so every point is weakly R connected to itself.

Suppose a function `tc :: Ord a => Rel a -> Rel a` for the transitive closure of a relation and a function `inv :: Ord a => Rel a -> Rel a` for inverting a relation are given. Use these to define a function

```
wConnected :: Ord a => Rel a -> a -> a -> Bool
wConnected r x y = ...
```

- $x == y$ || $elem\ (x,y)\ (tc\ (r\ ++\ (inv\ r)))$
- $x = y$ v $(x,y) \in (R \cup R^{-1})^{+}$

Jan van Eijck, *Paper Exam Software Specification and Testing*, 21 October 2013.

PROBLEM:

not enough practice

Student evaluation

- Unfortunately, the assessment was totally Haskell based, whereas we also learned a lot of non-Haskell material.
- we could have spent our time on also other important topics in software testing instead of logic and learning haskell
- I expected to learn more about software testing and this course was about logic
- Too much freaky math, but no real-life problems.
- There is no point at which this course connects to my professional career.
  Doing math-stuff with math-problems is the opposite of interesting.

PROBLEM:

the link is not apparent

Polar evaluation

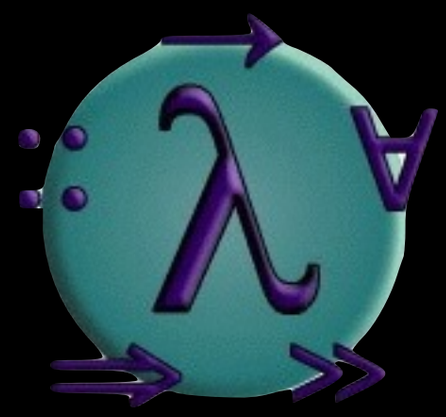
- I found learning Haskell on my own very difficult.
- The study load was very high due to the requirement to learn the Haskell language ourselves.
- [study load] WAY TOO HIGH
- The study load for me was low, because I used Haskell before
- The lab assignements were of good quality and fun to do.
- Really nice assignments! (Although the last one could have been made a bit more challenging...)

PROBLEM:

heterogeneous background

What was good?

- Appreciated parts:
  - modern technology
  - well-designed assignments
  - complementary reading
  - feedback on code [“late and brief”]
  - q&a sessions



Simon Oxley, Original Octocat, logo.

CAN WE
save the good parts
& improve the rest?

YES WE CAN

- Premaster course on FP
- Flipped classroom
- Guest lectures
- Integration with the paper sessions
- Hack sprints & competitive exercises

What is FLIPPED EDUCATION?

- (cf. Education Freedom Day 2014)
- Lecture & homework elements are reversed
- “Sage on the stage” $\Longrightarrow$ “guide on the side”
- Known since 199x, popular in 201x
- Claimed better use of class time
- Not a silver bullet

7 Things You Should Know About the Flipped Classroom (EDUCAUSE, 2012, CC-BY-NC-ND)
Vadim Zaytsev, Flipped Education, Education Freedom Day 2014.

Classic education

	Students	Instructor
Before Class	Homework (Reading §§)	“Homework” (Prep)
In Classroom	No Idea	Assume Usability
During Class	Follow	Get Through
After Class	Homework (Assignments)	“Homework” (Grading)
Away	Request Confirmation	Repeat

WHAT is the Flipped Classroom? (University of Texas at Austin)

Flipped education

	Students	Instructor
Before Class	Learn & Answer Questions	“Homework” (Prep)
In Classroom	Specific Questions	Anticipate Questions
During Class	Practice Skills Being Learnt	Guide With Feedback
After Class	Continue To Practice	Post Additional Info
Away	Seek Help When Needed	Continue To Guide

WHAT is the Flipped Classroom? (University of Texas at Austin)

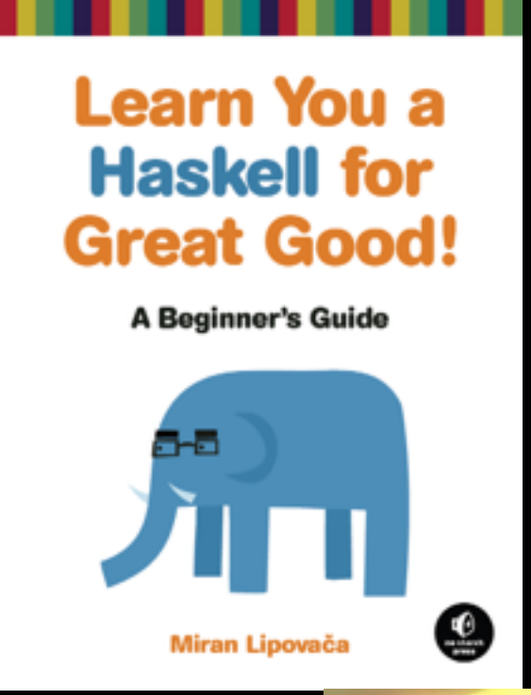
Expectations

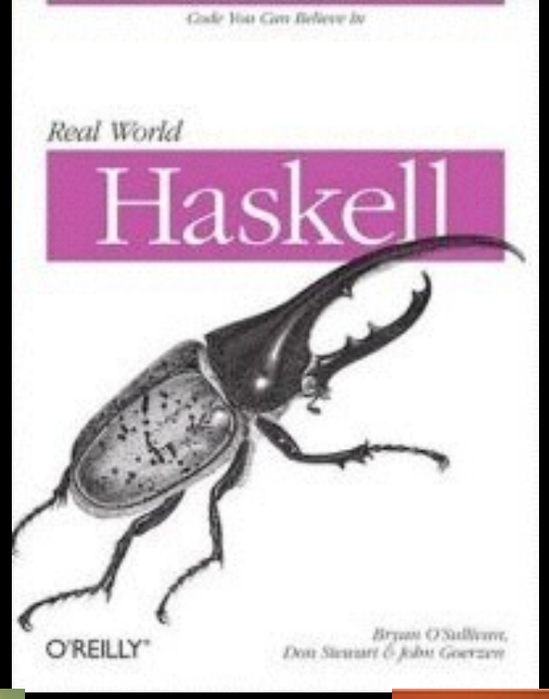
- Each student learns at her/his pace
  - never proceeding without mastery
- Better contact with students
  - “lecture” deals with their questions, not our expectations
- Selective & extensive exemplification
  - understand from 0, 1, …, 20 examples
- Analytics & diagnostics for further improvement

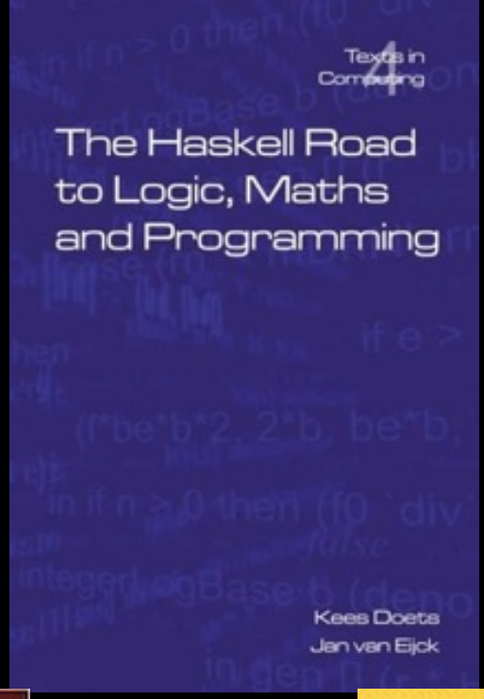
Redesign summary

- Split each lecture into separate smaller topics
- Per consumable topic,
  - §§, video clips
  - multiple examples
  - microtests for self-assessment
  - open questions to discuss later in class
- In lab assignments,
  - mostly the same structure
  - clearly identified goal for competition
  - occasional hack-sprints
- Presentations: each student reports about one testing method

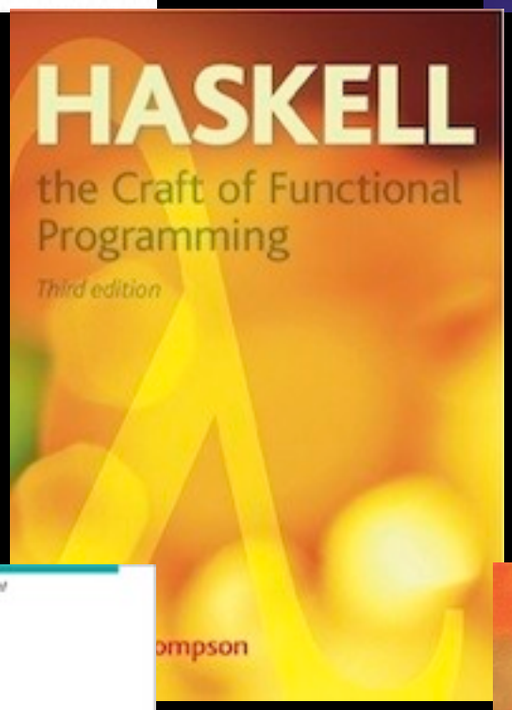
+Ask-Elle?

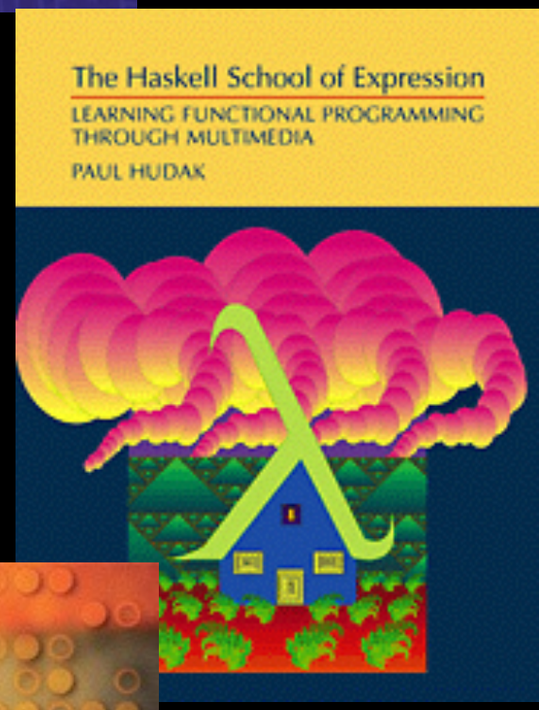

Learn You a Haskell for Great Good!
A Beginner's Guide
Miran Lipovača


Code You Can Believe In
Real World
Haskell
O'REILLY
Bryan O'Sullivan, Don Stewart & John Goerzen

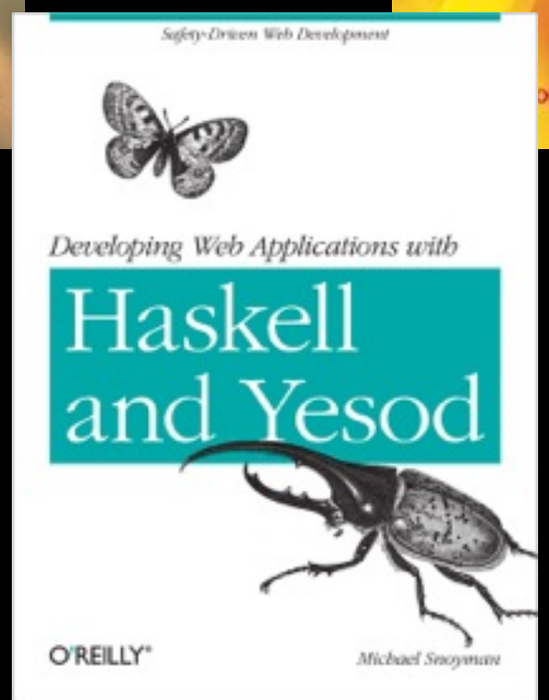

Texts in Computing
The Haskell Road to Logic, Maths and Programming
Kees Doets
Jan van Eijck

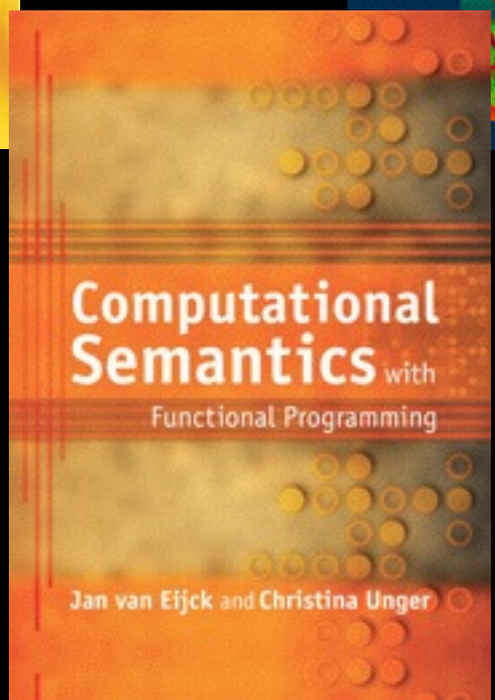

Graham Hutton
Programming in Haskell
CAMBRIDGE


HASKELL
the Craft of Functional Programming
Third edition
ompson


The Haskell School of Expression
LEARNING FUNCTIONAL PROGRAMMING THROUGH MULTIMEDIA
PAUL HUDAK


If you play around with Haskell, do not merely write toy programs.
Haskell Tutorial for C Programmers
OPENLIBRA
Eric Etheridge


Safety-Driven Web Development
Developing Web Applications with
Haskell and Yesod
O'REILLY
Michael Snoyman


Computational Semantics with Functional Programming
Jan van Eijck and Christina Unger

Conclusion

- Teaching Haskell at graduate level
  - for advanced software engineers
- Flipped classroom + contests + guest lectures
- More details in the full paper, also ask Jan & Vadim.
- First run in Sep-Oct 2014.
- Thank you for attention and feedback!
- Din Alternate Bold font by Linotype Library GmbH.
- Slides are CC-BY-SA: grammarware.net/talks/#TFPIE2014

Yanjing Wang, _20_0429, Feb 2007.