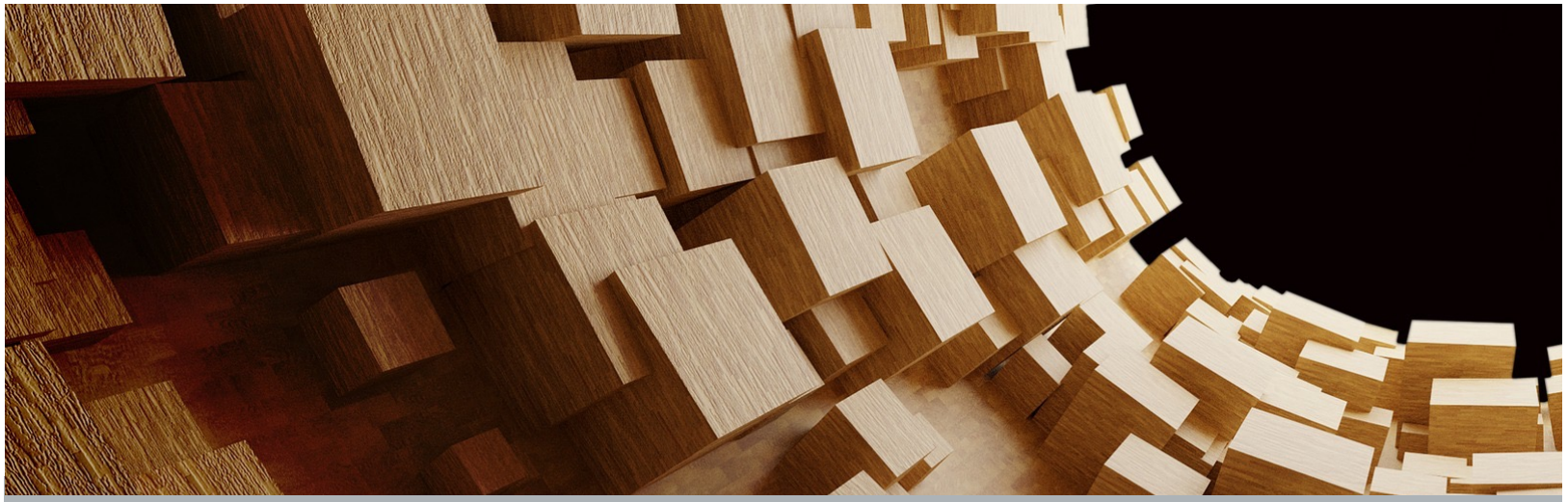
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT
FB Informatik, Programmiersprachen und Softwaretechnik

Extensibility and Modularity in Programming Languages

Seminar, WS 2017/18

17.10.2017 | Kick-off meeting



Introduction

Basic research questions

How can we design and/or use programming languages best to **extend** a program:

- in a **safe** way (types!)
- without **modifying** existing code

More generally: How can we best group a program into **modules** and how can this be supported by the programming language/ system?

A little example (Java)

```
interface Expression {
  int evaluate();
  String prettyPrint();
}
class Literal implements Expression {
  private int i;
  public Literal(int i) { this.i = i; }
  public int evaluate() {
    return i;
  }
  public String prettyPrint() {
    return i+"";
  }
}
```

A little example (Java)

Q: How can I add another kind of expression?

A little example (Java)

Easy:

```
interface Expression {
  int evaluate();
  String prettyPrint();
}
class Literal implements Expression { … }
class Addition implements Expression {
  private Expression e1, e2;
  … /* Constructor omitted */
  public int evaluate() {
    return e1.evaluate() + e2.evaluate();
  }
  public String prettyPrint() {
    return e1.prettyPrint() + "+" +
      e2.prettyPrint();
  }
}
```

A little example (Java)

Q: How can I add another kind of operation on expressions?

A little example (Java)

This is not quite so easy, though there are a number of attempts to solve this problem: Visitor pattern, Object algebras, …
(We'll learn more about them in the course of this seminar.)

However, it is easy to add new operations in most FP languages, like Haskell.

A little example (Haskell)

Q (for those who know Haskell):
How would you realize our little example in Haskell?

A little example (Haskell)

```
data Expr = Lit Int | Add Expr Expr

eval :: Expr -> Int
eval (Lit i) = i
eval (Add e1 e2) = (eval e1) + (eval e2)

pp :: Expr -> String
pp (Lit i) = show i
pp (Add e1 e2) =
  (show e1) ++ "+" ++ (show e1)
```

A little example (Haskell)

Adding a new operation is easy:

```
data Expr = Lit Int | Add Expr Expr

eval :: Expr -> Int
eval (Lit i) = i
eval (Add e1 e2) = (eval e1) + (eval e2)

pp :: Expr -> String
pp (Lit i) = show i
pp (Add e1 e2) =
  (show e1) ++ "+" ++ (show e1)

depth :: Expr -> Int
depth (Lit i) = 0
depth (Add e1 e2) =
  (maximum [depth e1, depth e2]) + 1
```

A little example (Haskell)

But: It is not so easy to add new variants of expressions — the situation is quite the opposite from Java.

Again there are a number of attempts to solve this problem for FP languages like Haskell: Datatypes à la carte, …
(We'll learn more about them in the course of this seminar.)

Expression Problem

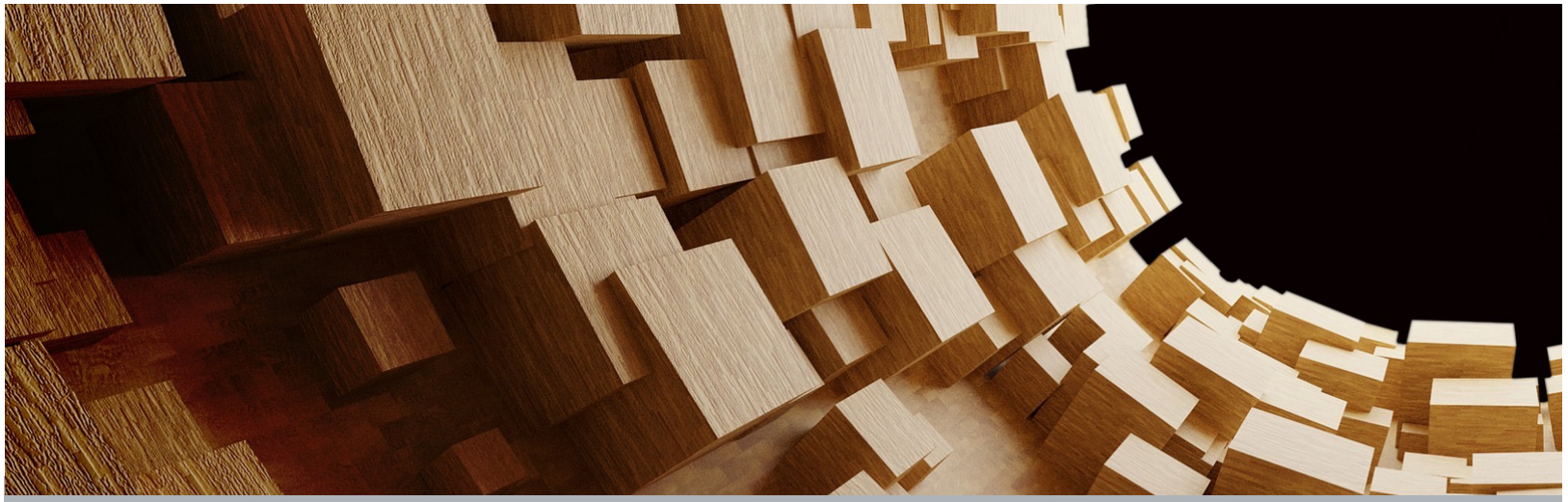
What we saw is the canonical example for what is now called the

Expression Problem *(name due to P. Wadler)*

How can we easily extend programs along ***both of the extensibility dimensions***: with both variants and operations ?

There are many variations on the problem itself and on the **conditions under which we consider it solved**:
type safety, various no-modification conditions, …

In this seminar: We study this matter in depth by reading some relevant **research papers**.



Organizational matters

Credit Points, Structure, Grading, Time slot

Credit Points

- 3 ECTS for M.Sc. module INFO4244
- 4 ECTS under old PO 2010 (as Pflichtseminar)

Structure of course

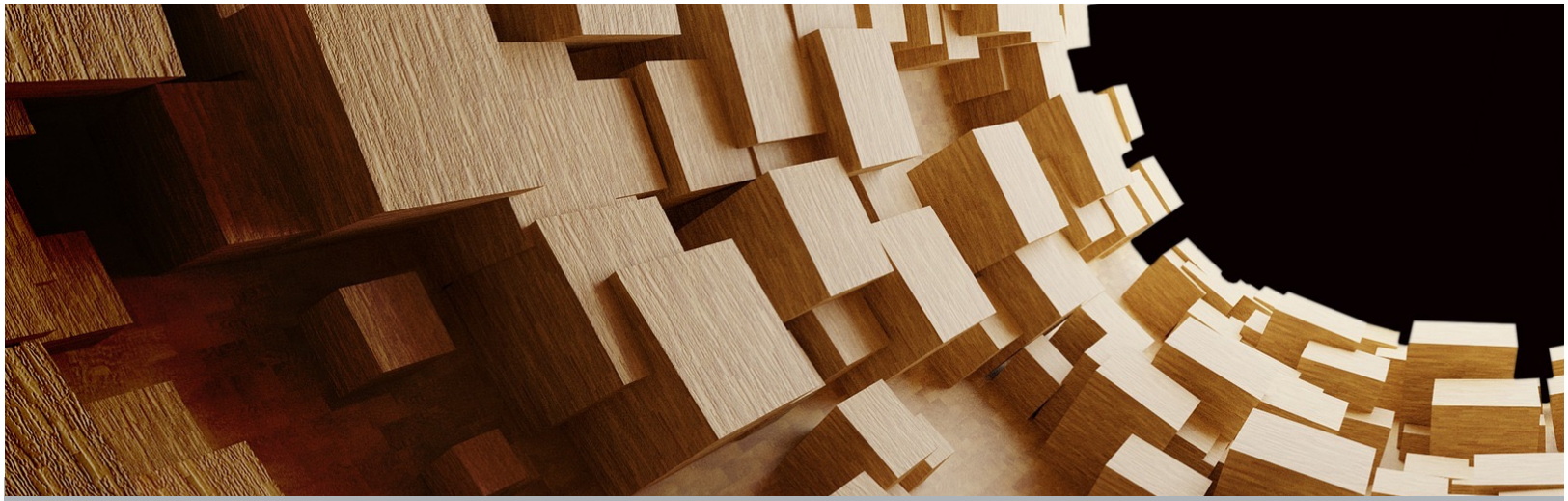
- Structure: Paper reading group with weekly meetings
- Each week: One student is the *discussion leader* who
  - picks a research paper,
  - familiarizes himself/herself in depth with its contents,
  - prepares for the discussion and questions, and
  - during the discussion: leads through the paper and keeps discussion on track.
- The others also read the paper and prepare questions, and
  - send these in due time before the meeting to me and the discussion leader.
- At the end of the semester: Each participant writes a term paper on the topic he prepared for as discussion leader.
  - More information on that will follow during the semester.

Grading

- 25% Participation as discussion leader
- 25% Participation in the other meetings
- 50% Term paper

Weekly meeting time slot?

- Ideally this time slot (Tuesday 16 c.t.-18), but we can try to find a better one that works for all.



How to read papers

How to read a CS research paper?

- Following P.W.L. Fong:
  http://faculty.ksu.edu.sa/chikh/Documents/reading-paper.pdf
- When trying to first comprehend the paper, answer these q's:
  - What is the research problem that is addressed?
  - What are the claimed contributions?
  - How do the author(s) substantiate these claims?
  - What are the conclusions? (What have we learned? What are open problems?)
- A paper can be seen as telling a *story,* and its *plot* is structured by these four questions.
- To then evaluate the paper, ask these questions:
  - Is the research problem significant?
  - Are the contributions significant?
  - Are the claims valid?

More information

More general information on reading papers:

- http://groups.csail.mit.edu/netmit/wordpress/wp-content/themes/netmit/papers/HowtoRead.pdf

Thank you.

Contact:

Julian Jabs
B221
Sand 13, 72076 Tübingen
julian.jabs@uni-tuebingen.de