FUNCTIONAL PROGRAMMING IN SCALA

The first steps to better code

WHOAMI

- Mauro Palsgraaf
- Graduating in 3 weeks
- Software developer
- FP believer

TABLE OF CONTENTS

- What is functional programming
- Why functional programming matters
- Functional programming fundamentals
- Freestyle
- References

WHAT IS FUNCTIONAL PROGRAMMING?

- Functions as building blocks
- Purity
- Referentially transparant

ADD(1, 3)

```
def add(x: Int, y: Int): Int = x + y
```

ISEVEN(**ADD(1, 3)**) == ISEVEN(**4**)

```
def add(x: Int, y: Int): Int = {
    println("adding x to y")
    x + y
}
```

ISEVEN(ADD(1, 3)) =/= ISEVEN(4)

Because we miss the println if we substitute

WHY IS RT VERY AWESOME?

IMPERATIVE PROGRAMMING

```
def buyCoffee(cc: CreditCard, p: PaymentService): Coffee = {

    val cup = new Coffee()

    p.charge(cc, cup.price)

    cup
}
```

IMPERATIVE PROGRAMMING

```
def buyCoffees(n: Int, cc: CreditCard, p: PaymentService): List[Coffee] = {
  var coffees: List[Coffee] = List()

  for (a <- 1 to n) {
    // prepend the new coffee to the list
    coffees = buyCoffee(cc, p) :: coffees
  }

  coffees
}
```

WHAT'S THE PROBLEM?

SIDE-EFFECTS ARE INTERMINGLED

FUNCTIONAL PROGRAMMING

```
def buyCoffee(cc: CreditCard): (Coffee, Charge) = {
    val cup = new Coffee()

    (cup, Charge(cc, cup.price)
}
```

FUNCTIONAL PROGRAMMING

```
def buyCoffees(n: Int, cc: CreditCard) : (List[Coffee], Charge) = {
    val purchases: List[(Coffe, Charge)] = List.fill(n)(buyCoffee(cc))

    val (coffees, charges): (List[Coffee],  List[Charge]) = purchases.unzip

    (coffees, charges.reduce((c1,c2) => Charge(c1.cc, c1.amount + c2.amount)))
}
```

AND AT THE VERY EDGE OF OUR PROGRAM

```
def charge(charge: Charge): IO[Unit] =
```

COMPOSITION & REUSABILITY

REASON ABOUT OUR SOFTWARE AS DATA TRANSFORMATION

WHY FUNCTIONAL PROGRAMMING MATTERS?

DOI:10.1145/3126905

A Large-Scale Study of Programming Languages and Code Quality in GitHub

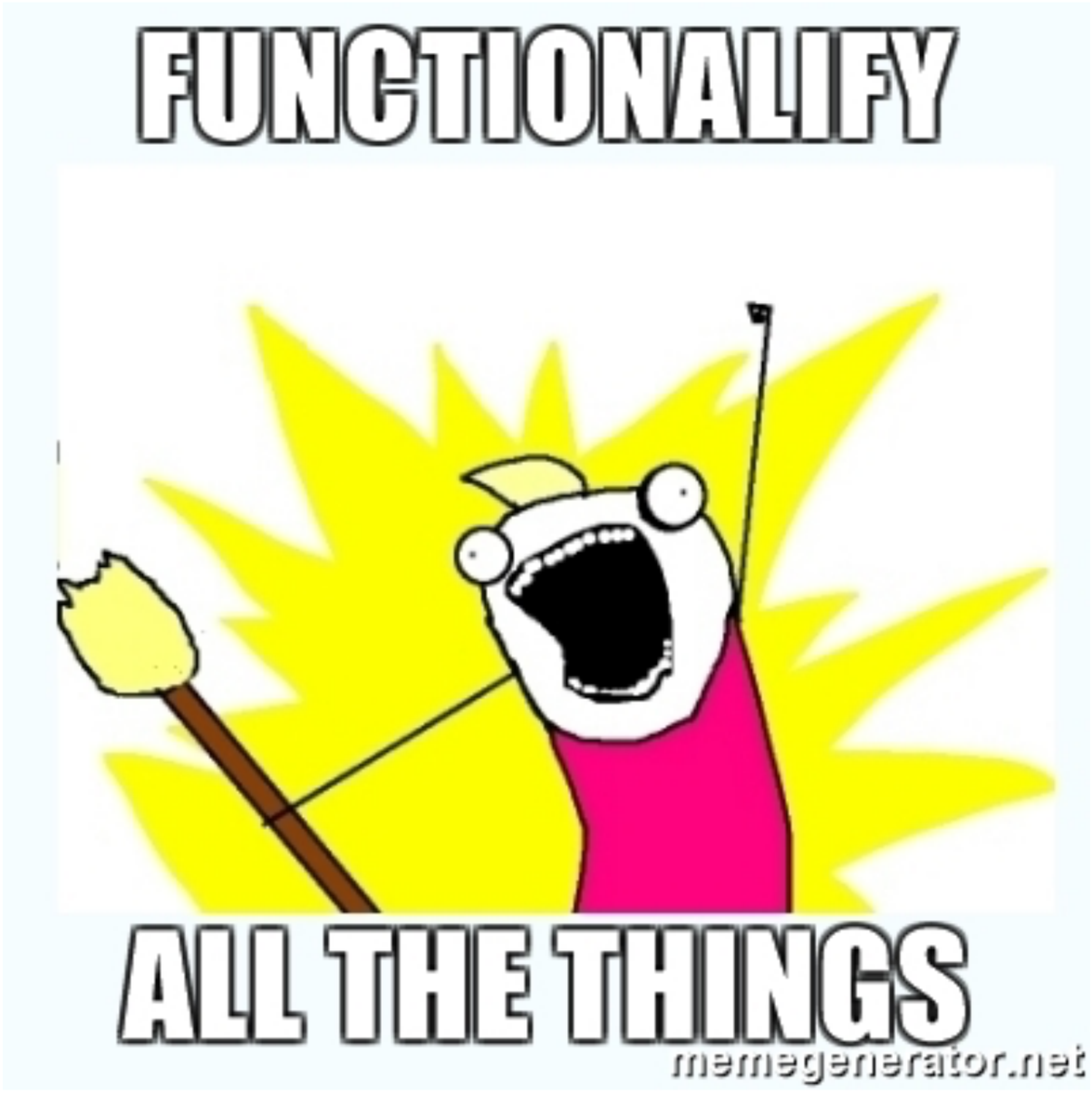
By Baishakhi Ray, Daryl Posnett, Premkumar Devanbu, and Vladimir Filkov

6. CONCLUSION

We have presented a large-scale study of language type and use as it relates to software quality. The Github data we used is characterized by its complexity and variance along multiple dimensions. Our sample size allows a mixed-methods study of the effects of language, and of the interactions of language, domain, and defect type while controlling for a number of confounds. The data indicates that functional languages are better than procedural languages; it suggests that disallowing implicit type conversion is better than allowing it; that static typing is better than dynamic; and that managed memory usage is better than unmanaged. Further, that the defect proneness of languages in general is not associated with software domains. Additionally, languages are more related to individual bug categories than bugs overall.

On the other hand, even large datasets become small and insufficient when they are sliced and diced many ways simultaneously. Consequently, with an increasing number of dependent variables it is difficult to answer questions about a specific variable's effect, especially where variable interactions exist. Hence, we are unable to quantify the specific effects of language type on usage. Additional methods such as surveys could be helpful here. Addressing these challenges remains for future work.

The data indicates that functional languages are better than procedural languages; it suggests that disallowing implicit type conversion is better than allowing it; that static typing is better than dynamic; and that managed memory usage is better than unmanaged.


FUNCTIONALIFY
ALL THE THINGS
memegenerator.net

WORKSHOP

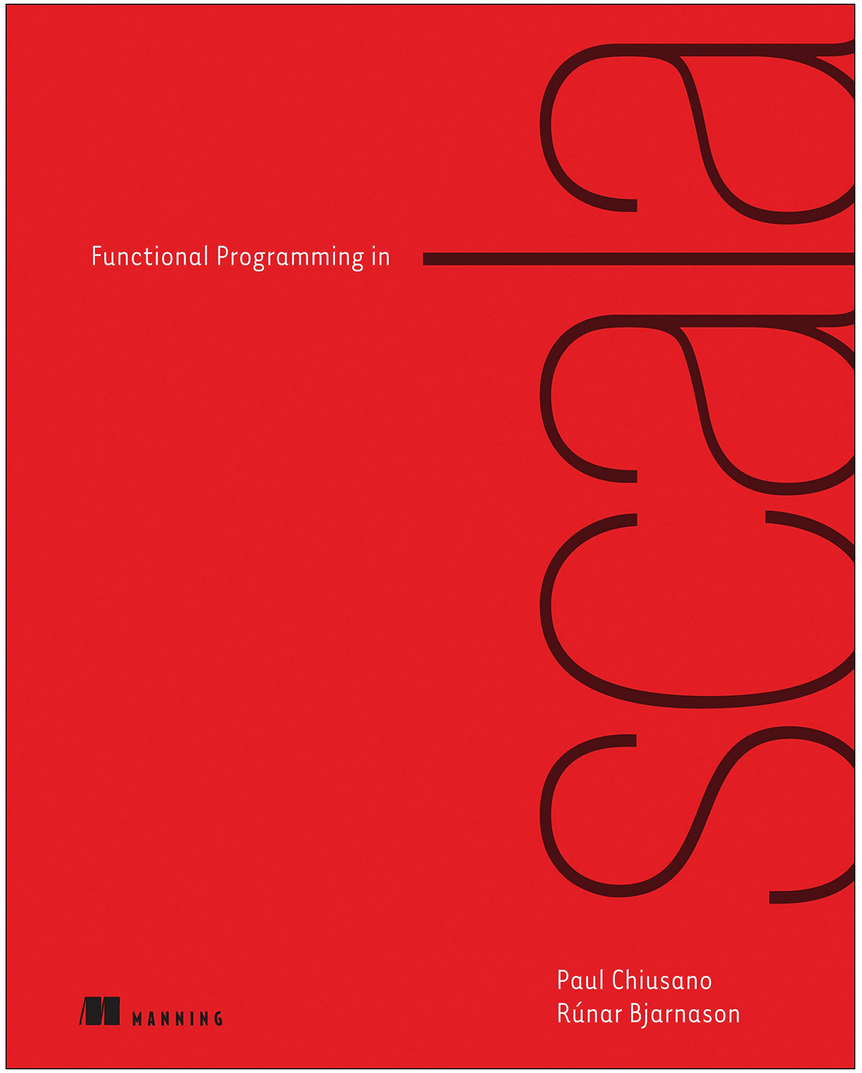
SUBJECTS

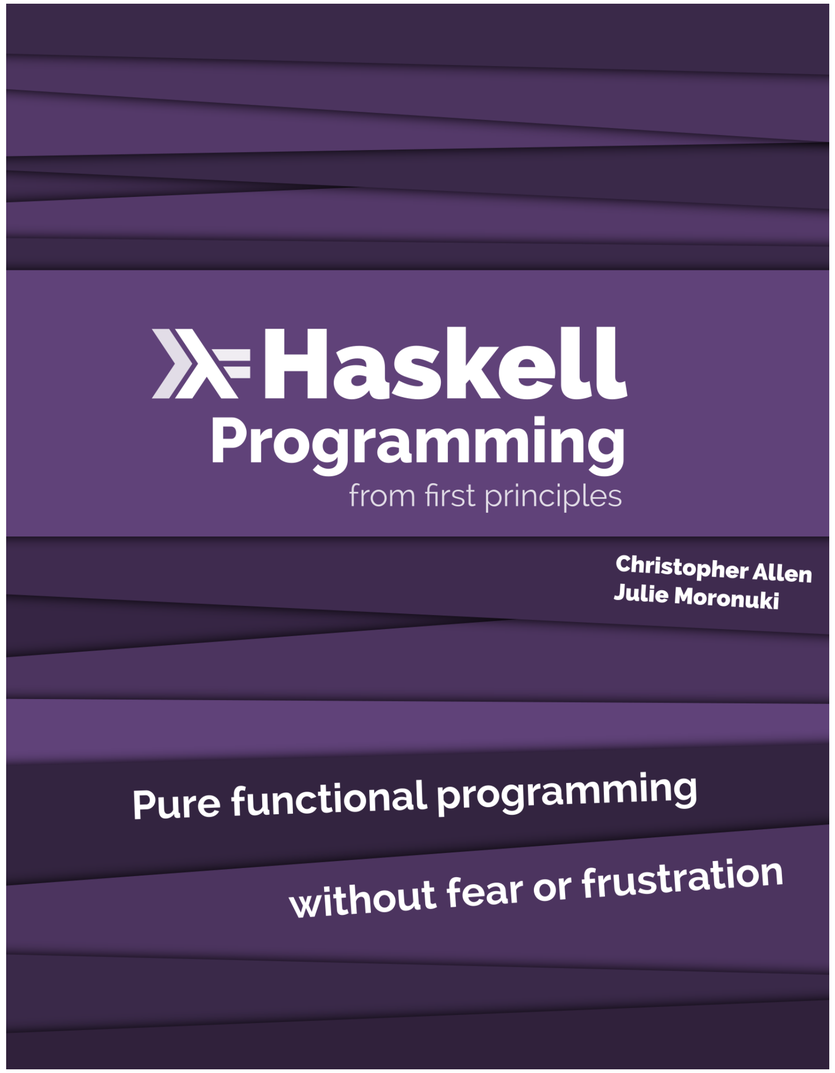
- ADTs
- Pattern matching
- Recursion / Tail recursion / folding
- Typeclasses

https://github.com/mauropalsgraaf/fpscala-workshop

FREESTYLE-
TIME

FUNCTIONAL PROGRAMMING IS A THING


Functional Programming in
Scala
MANNING
Paul Chiusano
Rúnar Bjarnason


Haskell
Programming
from first principles
Christopher Allen
Julie Moronuki
Pure functional programming
without fear or frustration

TWO MUST-READ PAPERS

- https://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf
- https://cacm.acm.org/magazines/2017/10/221326-a-large-scale-study-of-programming-languages-and-code-quality-in-github/fulltext