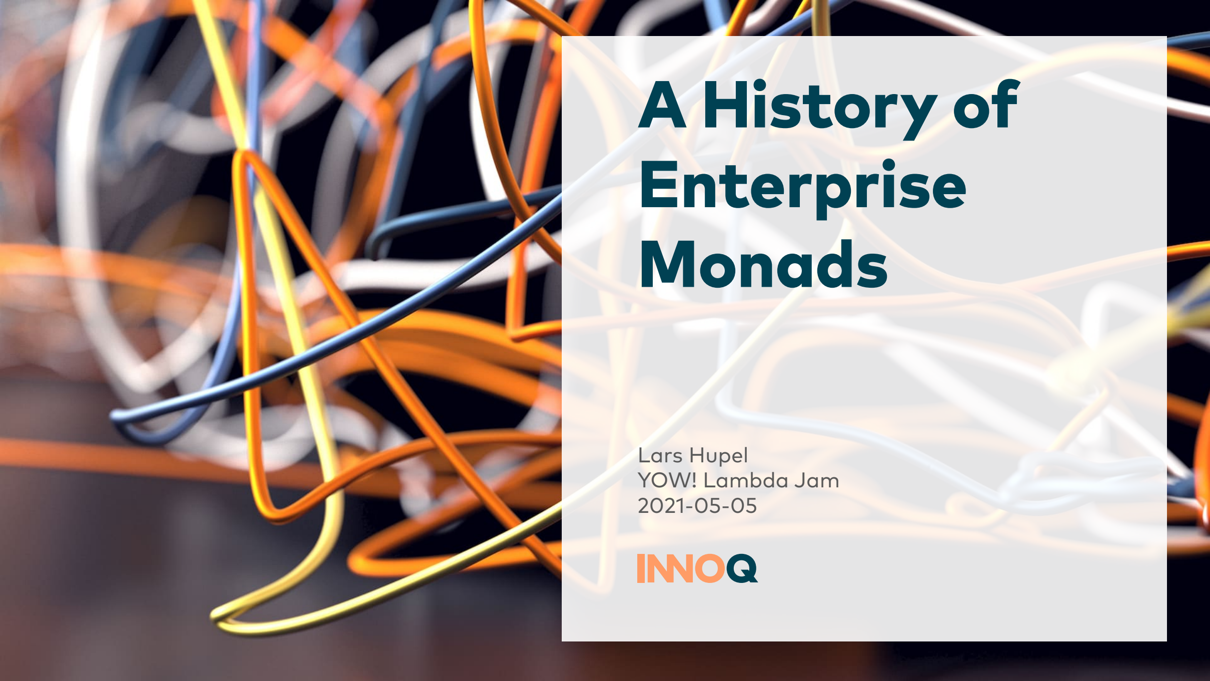


# A History of Enterprise Monads

Lars Hupel
YOW! Lambda Jam
2021-05-05

INNOQ

FLAMMABLE
KEEP FIRE AWAY
INFLAMMABLE
CAUTION
DELL

FLAMMABLE
KEEP FIRE AWAY
INFLAMMABLE
DELL

## Humble beginnings

Founded in 2013 at Northeast Scala Symposium

TYPELEVEL
SCALA

## Humble beginnings

Founded in 2013 at Northeast Scala Symposium

Today: 70+ projects, vibrant ecosystem

TYPELEVEL
SCALA

## Typelevel projects

Central theme: Scala-idiomatic Functional Programming

## Typelevel projects

Central theme: Scala-idiomatic Functional Programming

- … with as little hassle as possible

# Typelevel projects

Central theme: Scala-idiomatic Functional Programming

- … with as little hassle as possible
- … with as little runtime overhead as possible

## Typelevel projects

Central theme: Scala-idiomatic Functional Programming

- … with as little hassle as possible
- … with as little runtime overhead as possible
- … as safe as possible

## Adopters

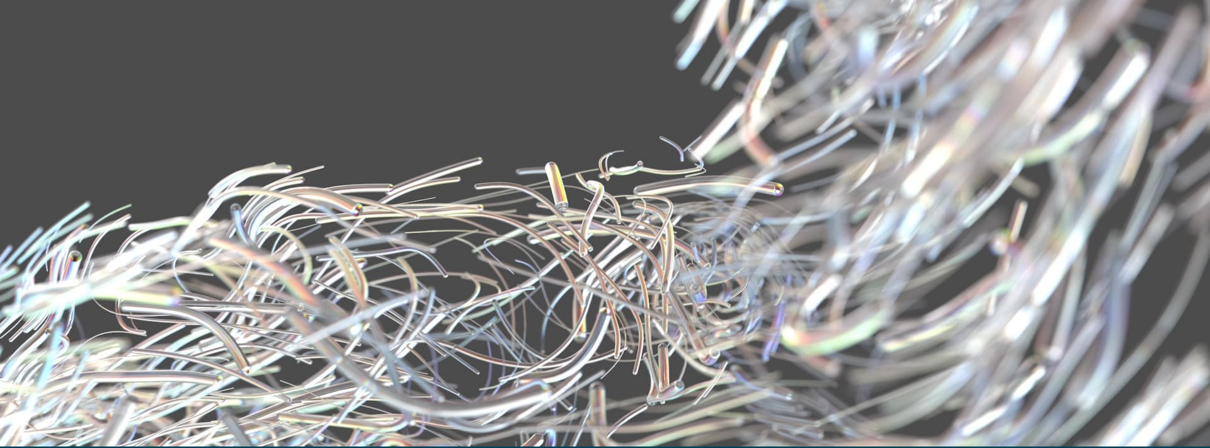
Disney streaming services

47

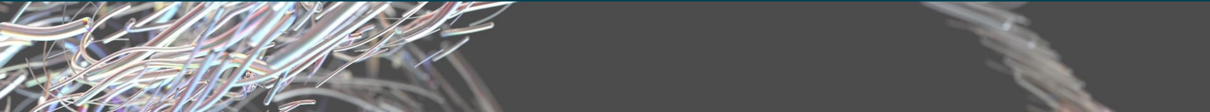
itv

COMCAST

PHILIPS



## Cats



## Type classes

Supremely useful tool, pioneered in Haskell

## Type classes

Supremely useful tool, pioneered in Haskell

```
class Semigroup a => Monoid a where
  mempty :: a
  mconcat :: [a] -> a
  mconcat = foldr mappend mempty
```

## Type classes

Supremely useful tool, pioneered in Haskell

```
class Semigroup a => Monoid a where
  mempty :: a
  mconcat :: [a] -> a
  mconcat = foldr mappend mempty
```

It Just Works™!

HULK SMASH THE STATE
hs
TYPELEVEL SCALA
Brendan McAdams
TYPELEVEL SCALA
Oslo Summit · May 4, 2016
Rickard Nilsson

## Type classes in Scala

… now we just need to encode them in Scala

## Type classes in Scala

… now we just need to encode them in Scala

- multiple inheritance?

## Type classes in Scala

… now we just need to encode them in Scala

- multiple inheritance?
- syntax??

## Type classes in Scala

… now we just need to encode them in Scala

- multiple inheritance?
- syntax??
- global confluence???

# Type classes in Scala

... now we just need to encode them in Scala

- multiple inheritance?
- syntax??
- global confluence???

# The Limitations of Type Classes as Subtyped Implicits (Short Paper)

Adelbert Chang
adelbertc@gmail.com

## Abstract

Type classes enable a powerful form of ad-hoc polymorphism which provide solutions to many programming design problems. Inspired by this, Scala programmers have striven to emulate them in the design of libraries like Scalaz and Cats.

The natural encoding of type classes combines subtyping and implicits, both central features of Scala. However, this encoding has limitations. If the type class hierarchy branches, seemingly valid programs can hit implicit resolution failures. These failures must then be solved by explicitly passing the implicit arguments which is cumbersome and negates the advantages of type classes.

In this paper we describe instances of this problem and show that they are not merely theoretical but often arise in practice. We also discuss and compare the space of solutions to this problem in Scala today and in the future.

the type class resolver automatically searches through the dictionary of instances to ensure the appropriate instances are defined.

Scala programmers have sought to emulate type classes to leverage this kind of ad-hoc polymorphism. The natural encoding of type classes uses implicits for instance definition and resolution and subtyping for specifying type class relationships.

As a running example consider the (stubbed) encoding of the `Functor` and `Monad` type classes. Each type class becomes a trait, and relationships between type classes become subtype relationships. For example, every `Monad` gives rise to a `Functor`, so `Monad[F] extends Functor[F]`.

```
trait Functor[F[_]] { }
trait Monad[F[_]] extends Functor[F] { }
```

It is also possible to write functions abstracting over these type classes.

## Type classes, encoded

In 2015, Michael Pilquist started *simulacrum*.

Goal: consistent encoding across different projects, 0 boilerplate

# Simulacrum

## Input

```scala
import simulacrum._

@typeclass trait Semigroup[A] {
  @op("|+|") def append(x: A, y: A): A
}
```

## Simulacrum

### Output

```scala
object Semigroup {
  def apply[A](implicit instance: Semigroup[A]): Semigroup[A] = instance

  // ...
}
```

## Simulacrum

### More output

```
object Semigroup {
  trait Ops[A] {
    def typeClassInstance: Semigroup[A]
    def self: A
    def |+|(y: A): A = typeClassInstance.append(self, y)
  }
}
```

# Simulacrum

## Even more output

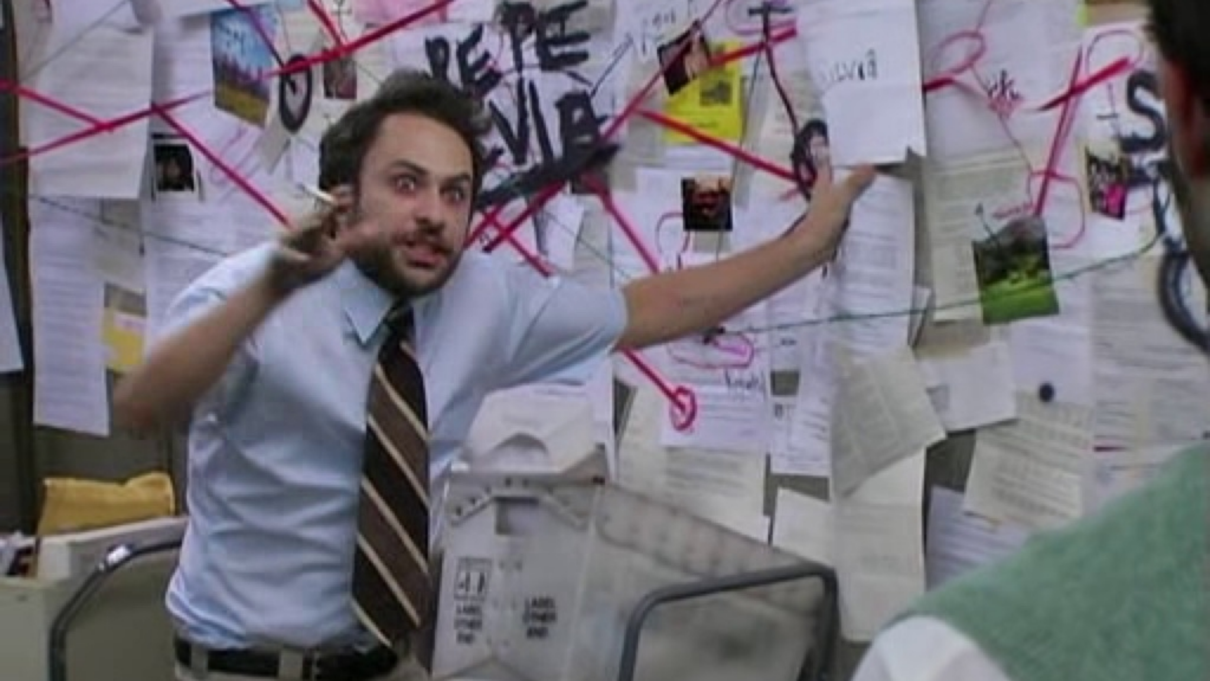
```scala
object Semigroup {
  trait ToSemigroupOps {
    implicit def toSemigroupOps[A](target: A)(implicit tc: Semigroup[A]): Ops[A]
      val self = target
      val typeClassInstance = tc
    }
  }

  object nonInheritedOps extends ToSemigroupOps
}
```

## Simulacrum

### Yet more output

```
object Semigroup {
  trait AllOps[A] extends Ops[A] {
    def typeClassInstance: Semigroup[A]
  }
  object ops {
    implicit def toAllSemigroupOps[A](target: A)(implicit tc: Semigroup[A]): AllO
      val self = target
      val typeClassInstance = tc
    }
  }
}
```


PEPE
SILVIA

## But it works!

Simulacrum solved a ton of issues

We can write `x |+| y`!

## But it works!

Simulacrum solved a ton of issues

We can write `x |+| y`!

Used by Cats and tons of third-party libraries

Spire

## Numerics for Scala

- started out as a SIP in 2011 (!)
- evolved into a dedicated library
- “what if functional but also fast”

## What about performance?

Simulacrum didn't solve the performance issue of type classes.

## What about performance?

Simulacrum didn't solve the performance issue of type classes.

**Input**

```
x |+| y
```

## What about performance?

Simulacrum didn't solve the performance issue of type classes.

**Output**

```
Semigroup.ops.toAllSemigroupOps(x).|+|(y)
```

## Enter Machinist

Split out of Spire by Erik Osheim in 2014

## Enter Machinist

Split out of Spire by Erik Osheim in 2014

Now (2020) archived and re-incorporated into Spire

## Shapeless

- started out as a series of talks in 2011 (!)
- scratched an itch:
  how to abstract over data?
- pioneered "type class derivation"
- many concepts incorporated into Scala 3

## Type Class Derivation

**Problem:** You want to serialize a bunch of case classes to JSON.

**Solution:** Boilerplate?

## Type Class Derivation

**Problem:** You want to compare a bunch of case classes.

**Solution:** Boilerplate … again?!

```scala
case class Account(owner: Person, balance: Int)

case class Person(name: String, address: Address)

case class Address(lines: List[String], country: Country)

case class Country(code: String)
```

```
type Account = Person :: Int :: HNil

type Person = String :: Address :: HNil

type Address = List[String] :: Country :: HNil

type Country = String :: HNil
```

## In Action

SCALA
CHECK

CIRCE

## Cats Effect

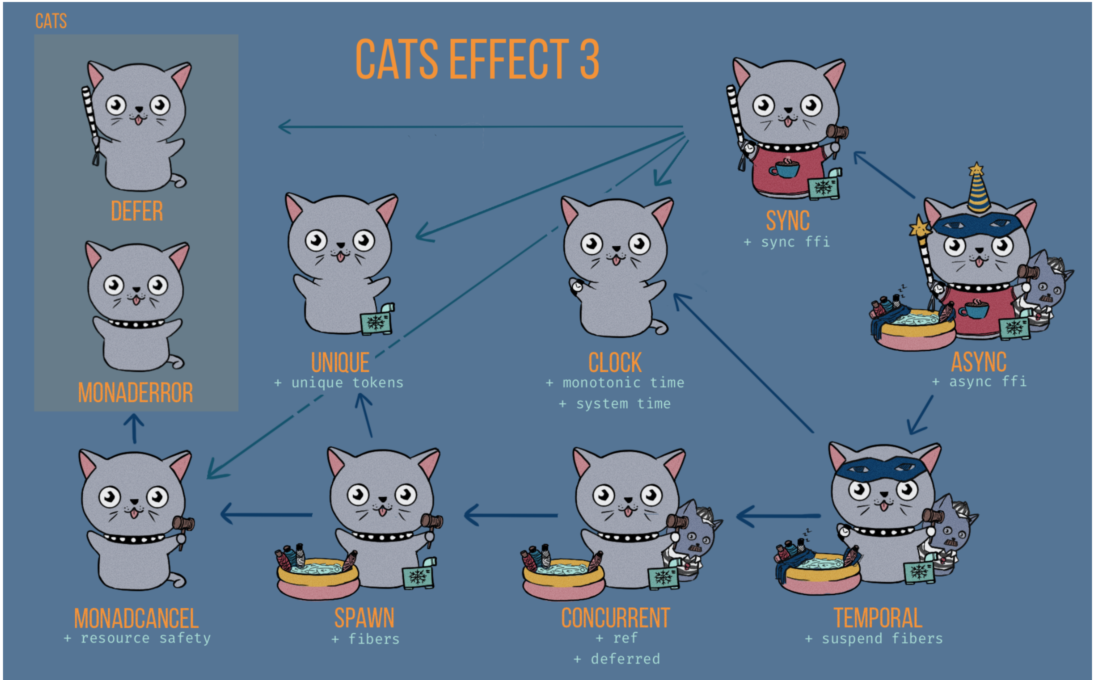
- full history almost impossible to trace
- draws from multitude of influences
- supports the rise of asynchronous software construction

Functional Programming in Scala

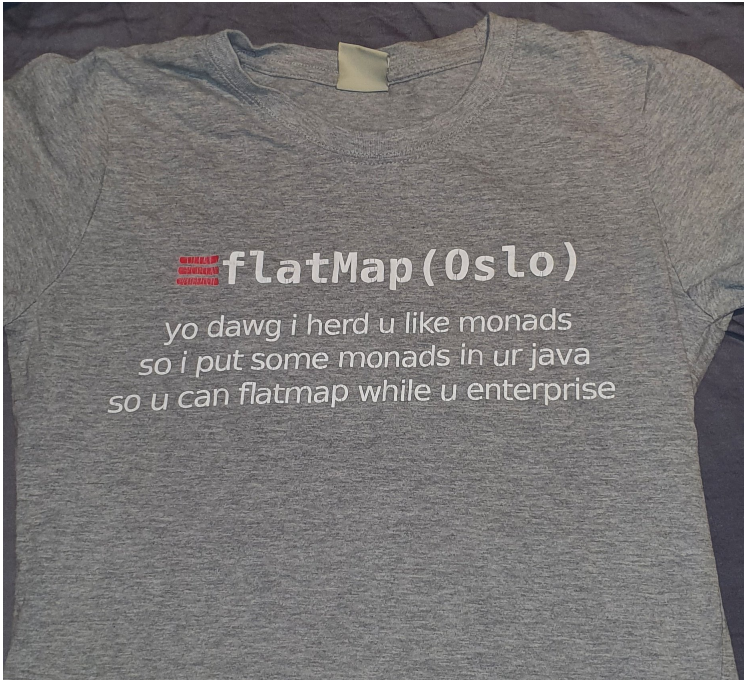
Paul Chiusano
Rúnar Bjarnason
Foreword by Martin Odersky

MANNING

# CATS EFFECT 3



## Future


flatMap(Oslo)
yo dawg i herd u like monads
so i put some monads in ur java
so u can flatmap while u enterprise

seriously the answer is almost always .traverse

So are we not flatmapping that shit any more?

traverse is flatmapping that shit on our behalf

## Q & A

INNOQ
www.innoq.com

Lars Hupel

lars.hupel@innoq.com

@larsr_h

# LARS HUPEL

**Senior Consultant**
**innoQ Deutschland GmbH**

Lars is known as one of the founders of the Typelevel initiative which is dedicated to providing principled, type-driven Scala libraries in a friendly, welcoming environment. A frequent conference speaker, they are active in the open source community, particularly in Scala.

## Sources

- https://pixabay.com/photos/people-business-meeting-1979261/
- https://unsplash.com/photos/FaNUdWGJqBg
- https://twitter.com/bodil/status/1383908807588204552/photo/1
- https://twitter.com/milessabin/status/1364523756601937921
- https://www.manning.com/books/functional-programming-in-scala
- https://impurepics.com/posts/2021-03-31-cats-effect-3.html
- https://twitter.com/tpolecat/status/721019769869045760