

Shape Safety in Tensor Programming is Easy for a Theorem Prover

Project: <https://github.com/tribbloid/shapesafe>

Content: <https://github.com/tribbloid/shapesafe-demo>

-- Peng Cheng - tribbloid@github | [twitter](#)

About Me



Maintainer of DataPassports, Computing Engine ... 2015 ~

- Only read Scala 'cause of Apache Spark
- Only read type theory 'cause math seems more stable than programming
- Not a Python fan, but only use PyTorch 'cause it is not TensorFlow

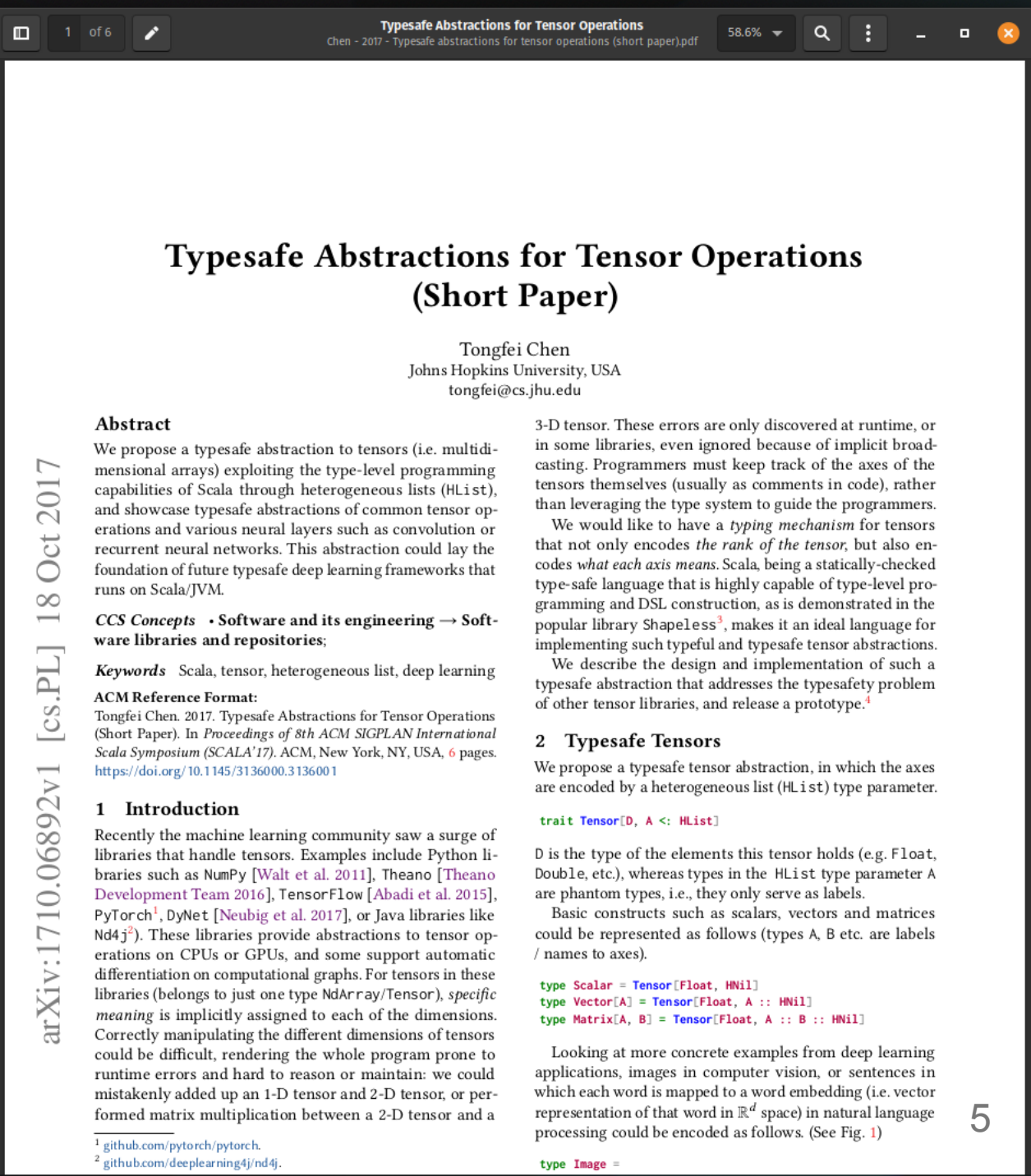
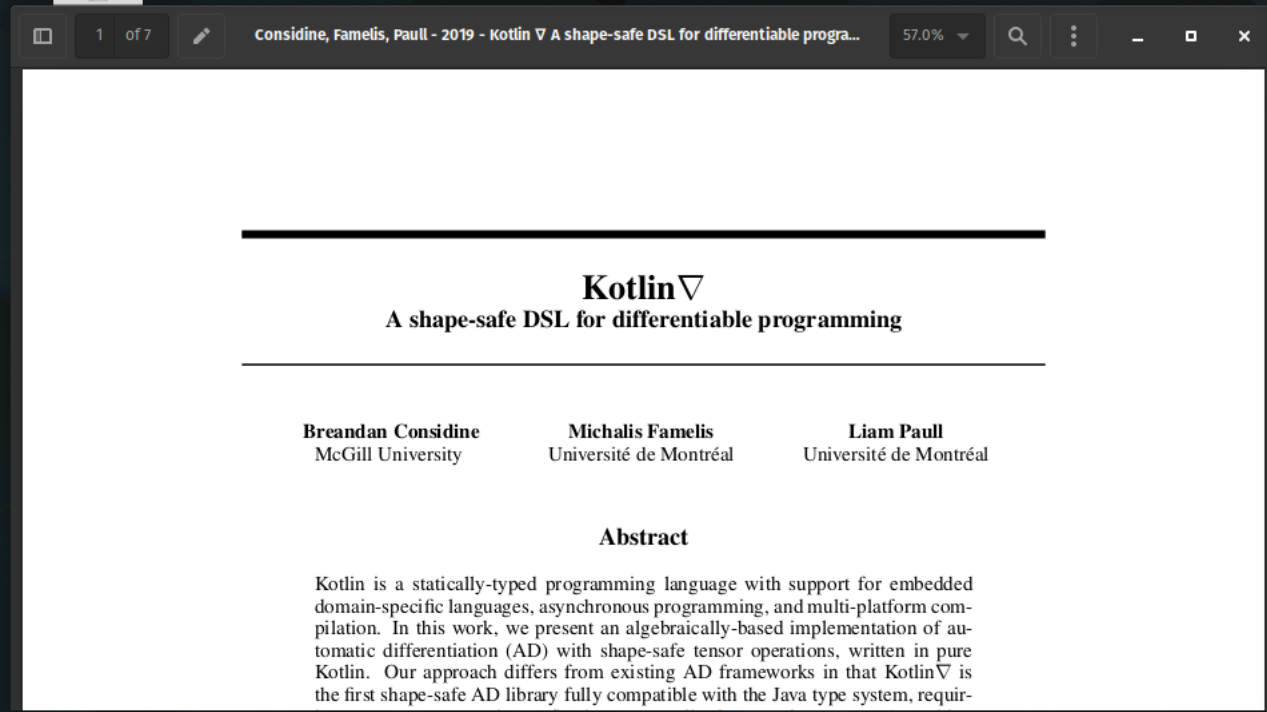
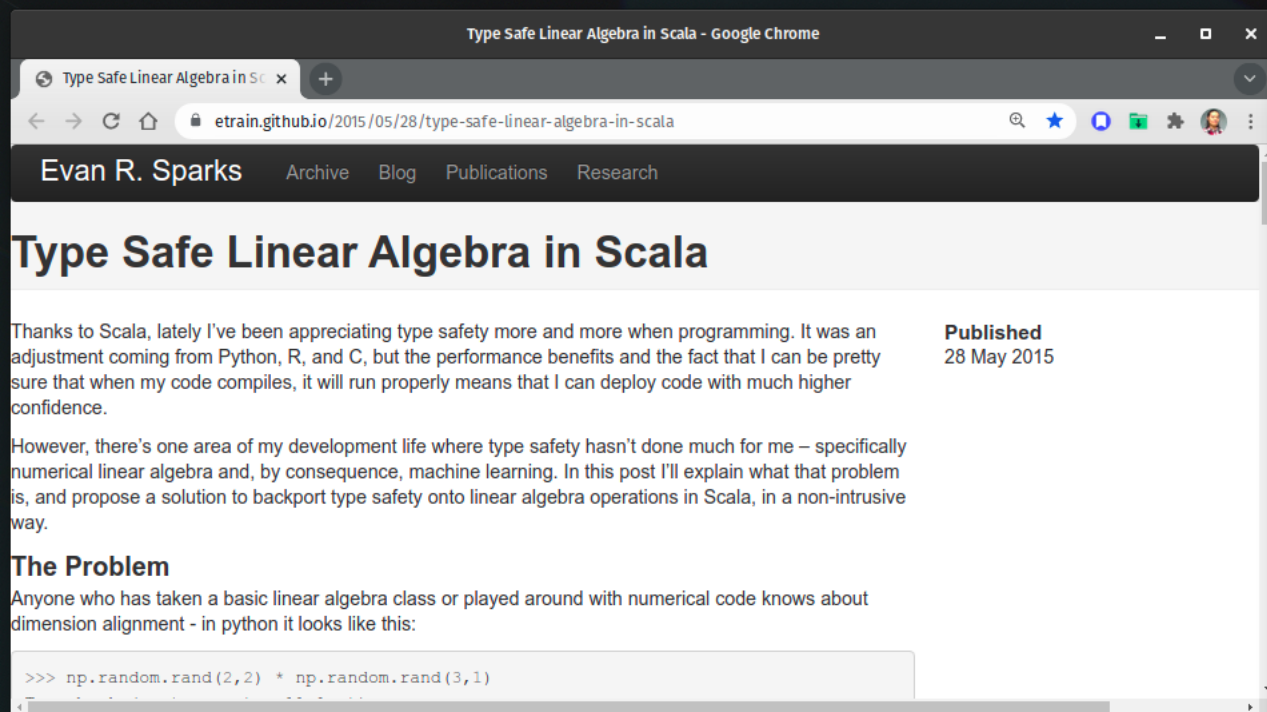
Maintainer of splain plugin ... 2021 ~

- Partially integrated with scala compiler 2.13.6+ `-Vimplicit` `-Vtype-diff`

Overview

- Why type-safe linear algebra?
- How to push it to extreme?
- Why scala?
- What's next?

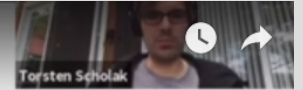
Why type-safe linear algebra?



arXiv:1710.06892v1 [cs.PL] 18 Oct 2017

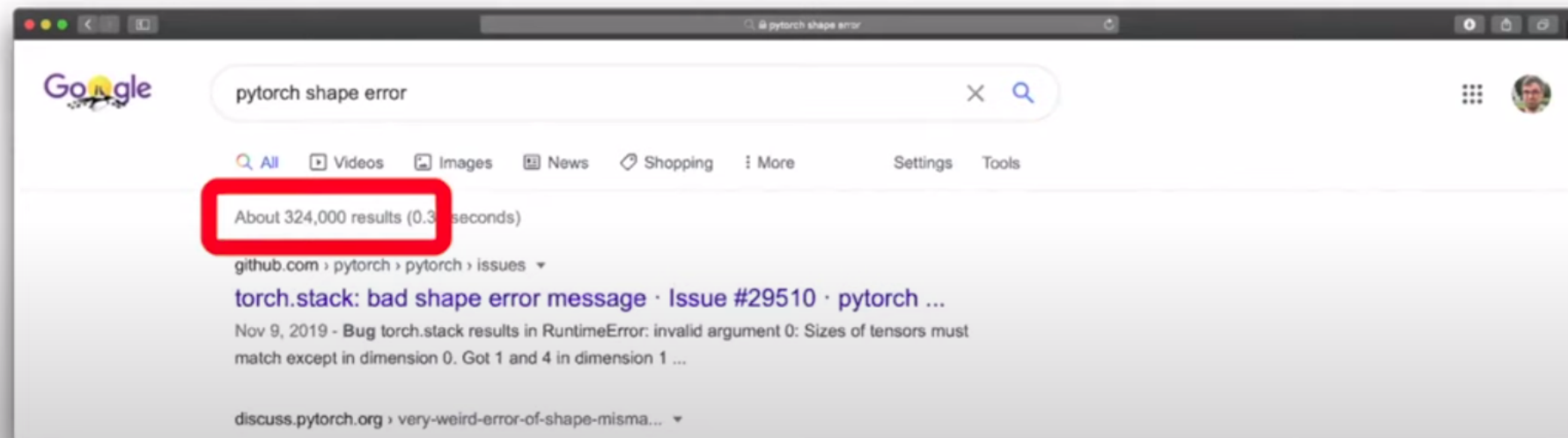
Designed For Human (who make mistakes)

FP Forum Online: Flavio Corpa, Torsten Scholak, and Richard Feldman



Type-safe Tensor Math

- Correctly manipulating tensors of different shapes is hard 😓
- In PyTorch, shape mismatch errors typically only discovered at runtime 😱



- In Hasktorch, tensor **shapes**, **dtypes**, and **devices** can be checked by GHC 🙌

OR is it?

18th July, 2021

ICML

Long talk

Neural Architecture Search Without Training

Joseph Mellor, Jack Turner, Amos Storkey, Elliot J. Crowley.

A low-cost measure for scoring networks at initialisation which allows us to perform architecture search in a matter of seconds instead of hours or days.



19th Apr, 2021

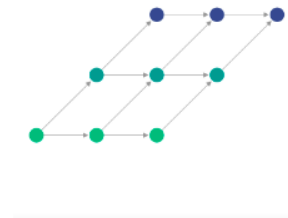
ASPLOS

Distinguished Paper

Neural Architecture Search as Program Transformation Exploration

Jack Turner, Elliot J. Crowley, Michael O'Boyle.

A compiler-oriented approach to neural architecture search which allows us to generate new types of convolution.



6th Dec, 2020

NeurIPS

Spotlight

Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels



- (source: <https://arxiv.org/abs/2102.06599>)

How to push it to extreme?

Curry-Howard(-Lambek) isomorphism

proof system	\iff	functional programming (<i>since 1930</i>)
Proposition P		<code>type P <: Any</code> (in MLTT) <code>type P <: Prop</code> (in CiC)
Proposition $P(a \in A)$		<code>type P[_ <: A] , a: {type P}</code>
Inductive Proposition $P(a_{i+1}, P(a_i))$		<code>type ::[HEAD, TAIL <: P] <: P</code>

... with quantifiers

proof system	\iff	functional programming (<i>since 1970</i>)
$\forall a \in A, P(a)$		<pre>def p[AS <: A with Singleton](a: AS): P[AS] , def p[AS <: A with Singleton](a: AS): AS#P , def p(a: A): a.P</pre>
$\exists b \in B, P(b)$		<pre>def p[B]: P {val b <: B, type P}</pre>
Axiom		<pre>def axiom[X](.): P[X]{.}</pre>
Theorem		<pre>def theorem[X](., lemma1: X => X#L1): P[X]{.}</pre>

... in shapesafe impl

proof system	\iff	functional programming (after η -expansion)
$\forall a \in A, P(a)$		<pre>def p[AS <: A with Singleton]: AS - P[AS] , def p[AS <: A with Singleton]: AS - AS#P</pre>
$\exists b \in B, P(b)$		<pre>def p[B]: P {val b <: B, type P}</pre>
Axiom		<pre>def axiom[X]: X - P[X]{.}</pre>
Theorem		<pre>def theorem[X](lemma1: X => X#L1): X - P[X]{.}</pre>

Scala 2.10 - 2015

```
class Matrix[A,B](val mat: DenseMatrix[Double]) {  
  def *[C](other: Matrix[B,C]): Matrix[A,C] = new Matrix[A,C](mat*other.mat)  
}  
  
class N  
class D  
class K  
  
val x = new Matrix[N,D](DenseMatrix.rand(100,10))  
val y = new Matrix[N,K](DenseMatrix.rand(100,2))  
  
val z1 = x * x //Does not compile!  
val z2 = x.t * y //Compiles! Returns a Matrix[D,K]
```

Scala 3.1 - 2021

```
import scala.compiletime.ops.int._

class Matrix[A, B]():

  def *[C](other: Matrix[B, C]): Matrix[A, C] = new Matrix[A, C]()
  def conv[C, D](other: Matrix[C, D]): Matrix[A - C + 1, B - D + 1] =
    new Matrix[A - C + 1, B - D + 1]()

val x = new Matrix[100, 100]()
val y = new Matrix[3, 3]()
val w = x.conv(y)

val z1 = w * new Matrix[100, 1]() //Does not compile!
val z2 = w * new Matrix[98, 1]() //Compiles!
```

Extreme 1

- Weird operands (Seriously, who is going to write compile-time type evaluation for einsum?)

```
type EinSum[Matrix[N1 -> D1, N2 -> D2]] = {  
  if (N1 == N2) {  
    if (D1 == D2) {  
      Matrix[N1 -> D1]  
    }  
    else {  
      error(...)   
    }  
  }  
  else {  
    Matrix[N1 -> D1, N2 -> D2]  
  }  
}
```

Extreme 2

- Symbolic reasoning (`p.apply` + `==` can only go in 1 direction)

```
class Matrix[A, B]():  
  
  def +(other: Matrix[A, B]): Matrix[A, B]  
  def transpose: Matrix[B, A]  
  def flatten: Matrix[A * B, 1]  
  
type A <: Int  
type B <: Int  
val m = new Matrix[A, B]()  
m.flatten + (m.transpose.flatten) // Oops
```



Learn from

- forward mode

```
lemma and_is_comm (p q: Prop) (h: p ∧ q): q ∧ p :=  
  and.intro (h.right) (h.left)
```

- tactic mode

```
lemma and_is_comm' (p q: Prop) (h: p ∧ q): q ∧ p :=  
begin  
  apply and.intro,  
  exact h.right,  
  exact h.left,  
end
```


Full auto mode

```
lemma and_is_comm' (p q: Prop) (h: p ∧ q): q ∧ p :=  
begin  
  assumption,  
  assumption,  
  assumption,  
end
```

==:

```
begin  
  assumption | ? => q ∧ p | .intro: (q, p)? => q ∧ p  
  assumption | p ∧ q => ? | .right: p ∧ q => p?  
  assumption | p ∧ q => ? | .left: p ∧ q => q?  
end
```

... in Scala!

soronpo Apr '19

odersky: Jul 2020

That syntax should be very different from normal parameters and arguments.
EDIT: In fact it's better not to think of them as parameters at all, but rather see them as *constraints* .

I always thought of them as (formal/mathematical) *assumptions*

odersky:

6. Imports of implicits should be clearly differentiated from normal imports.

Will there be a separate SIP for implied imports (or where should I comment on its semantics)?

Reply

(source: Oron Port's response to "Principles for Implicits in Scala 3" by Odersky)

Curry-Howard isomorphism, rewritten

proof system	\iff	functional programming (full auto!)
$\forall a \in A, P(a)$		<pre>def p[AS <: A with Singleton]: AS - P[AS] , def p[AS <: A with Singleton]: AS - AS#P</pre>
$\exists b \in B, P(b)$		<pre>def p[BS <: B](implicit b: BS): BS - BS#P</pre>
Axiom		<pre>def axiom[X]: X - P[X]{.}</pre>
Theorem		<pre>def theorem[X](implicit lemma1: X => X#L1): X - P[X]{.}</pre>


<Life Coding>

Why Scala

-- Specifically, why scala 2?


30 Answers


Active Oldest Votes


 Bryan Birch is credited with once saying that he programmed in a very high-level programming language called "graduate student".


186


Share Cite Edit Follow Flag answered Jan 8 '10 at 11:26 community wiki Kevin Buzzard




82  "Undergrad" is powerful enough for many common applications, and less resource-hungry. – Mark Meckes Jan 8 '10 at 14:40

111  It's pretty buggy though. – Allen Knutson Jan 8 '10 at 15:49

22  Lenstra liked to say that Joost Batenburg was an essential part of the "Bread and Beer" algorithm: "When we had a problem, we would put him in the computer with some bread and beer. A few hours later, he would emerge with a solution." – S. Carnahan Jan 8 '10 at 19:02

11  The use of "do" psuedocode in Hilbert's problems was pretty effective. – Steve Huntsman Jan 10 '10 at 17:39

6  I hope Lenstra at least installed an extra case fan. – Greg Marks Feb 1 '11 at 1:10

Add a comment | Show 1 more comment

Stack of AI-HPC-Hardware co-evolution

^ More Abstract

- Distributed Computing --- Apache {Spark | Flink}
- [*SOME Deep Learning Library?*]
- IR/Multi-stage Compilation --- reflection, LMS
- Hardware Design --- CHISEL / SpinalHDL

V More Concrete

But try to avoid ...

- Diverging implicit
 - Circular reference `def theorem[A](using a: A): A`
 - Expanding reference `def theorem[A <: F[_], B](using a: F[A, B]): A`
- Summoning proof on long algebraic data type

```
SquashByName[ / ]#On[
  SquashByName[ / ]#On[
    SquashByName[ + ]#On[
      SquashByName[ - ]#On[
        SquashByName[ / ]#On[
          SquashByName[ + ]#On[
            SquashByName[ - ]#On[
              SquashByName[ / ]#On[
                SquashByName[ + ]#On[
                  SquashByName[ - ]#On[
                    1024 >< 1024 |<<- (i >< j) >< (3 >< 3 |<<- (i >< j))...
```

What's next?

- Gradually typed (`UncheckedShape`), but still no symbolic reasoning
- Fairly stable, but not released or in production
- Not a computing library! Cannot empower ML engineers directly, but serve as a bridge to get there
- Need Scala 3 support!

Moving to Scala 3?

Pros

- Implicit search tree traversal is aggressively cached
- No more unpredictable diverging implicit recovery
- Theoretically sound resolving of `<::<` and `==::=`

Cons

- No type projection (may be added back)
- No shapeless Record type
(may be integrated into *Programmatic Structural Types*)
(can be expressed in match type in a very inefficient way)

The End Game?

"Scala" is going nowhere

"Scala is a gateway drug to ~~Haskell~~"
Agda

*Recognizing this fact,
we should phase out the name "Scala"*



Hascallator

Proudly Supported By

DataPassports™, the Data Privacy & Trust Company

- We are hiring! Looking for a director of engineering

Splain Open-Source Team

- Looking for scala compiler guru / type theorist as technical advisor
- 1.0.0RC is close, looking for test users