Pick your number type with Spire

Denis Rosset

physics@denisrosset.com
@denisrosset
github.com/denisrosset

$$ Perimeter Institute
$$ John Templeton Foundation

❤️ typelevel/spire

Pick your number type with Spire

What is a number ?

Integers, real numbers, fractions, …

Complex numbers, polynomials, …

Timestamps, time offsets, …

Pick your number type with Spire

How do we encode numbers on a computer ?

Operations (API)

Internal representation (precision, speed)

Printing and parsing

Pick your number type with Spire

Spire

Typelevel project (Erik Osheim, Tom Switzer & collabs)

Compat. with cats (cats-kernel) and twitter/algebird

8 years old, 33 kLOC

Version 0.16.0.2

Pick your number type with Spire

Spire

Precise API for numbers (type classes)

Generic programming (reduce boilerplate)

Data types

(Rational, Algebraic, Interval, Complex, Quaternion …)

What's wrong with the Scala stdlib?

Enables generic programming

```
trait IterableOnceOps[+A, +CC[_], +C] {

    def sum(implicit num: Numeric[A]): A =

        foldLeft(num.zero)(num.plus)

}


Seq(1,2,3).sum ; Seq(1.0,2.0,3.0).sum ; ...
```

What's wrong with the Scala stdlib?

Not precise

```
trait Numeric[T] extends Ordering[T]
```

What about complex numbers or coordinates? What are the semantics of quot/rem?

Boxes

```
trait Numeric[@specialized T] extends Ordering[T]
```

What are the laws?

Plan

- What are numbers and their representations?
- APIs for numbers and laws
- Performance across JVM, JS and Native

What are numbers, and how to encode them?

- Natural numbers 1,2,3,4,5…
- Signed numbers ..., -2, -1, 0, 1, 2, … (BigInt)
- Fractions p/q

  p is a signed number

  q is a natural number

  `(BigInt, BigInt)`
- Real numbers

  “*a real number is a value of a continuous quantity that can represent a distance along a line*”

Real numbers in mathematics

Construction of the real numbers

From Wikipedia, the free encyclopedia

In mathematics, there are several ways of defining the rea
numbers as a *complete ordered field*. Under the usual axi
for the axioms, and any two such models are isomorphic. A
basic properties of the rational number system as an orde

Contents [hide]

1 Synthetic approach
  1.1 Tarski's axiomatization of the reals
2 Explicit constructions of models
  2.1 Construction from Cauchy sequences
  2.2 Construction by Dedekind cuts
  2.3 Construction using hyperreal numbers
  2.4 Construction from surreal numbers
  2.5 Construction from Z (Eudoxus reals)
  2.6 Other constructions
3 See also
4 References

Real numbers on a computer

- Exactly computable subsets

  Rational numbers (+, -, *, /)
  Algebraic numbers (roots of polynomials, +, -, *, /, sqrt, cos(rat*pi))
  (hard problem in general)

- Approximations

  Decimal floating point: integer * 10^exponent (`BigDecimal`)
  Binary floating point: integer * 2^exponent (`Float`, `Double`)
  Expression trees

Why does it matter?

```
scala> Array.fill(10000)(0.1).sum - 1000

res3: Double = 1.5882051229709 76E-10


scala> Array.fill(10000)(0.25).sum - 2500

res4: Double = 0.0


scala> Array.fill(10000)(BigDecimal(0.1)).sum - 1000

res5: scala.math.BigDecimal = 0.0
```

The JVM, Scala and Spire

- JVM primitive types

Int	32 bits	−2,147,483,648 to 2,147,483,647
Long	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	32 bits	6-9 decimal digits, 10^-38 to 10^38
Double	64 bits	15-17 decimal digits, 10^−308 to 10^308

- JVM reference types

BigInteger	(arbitrary range)
BigDecimal	(arbitrary range, MathContext for decimal digits)
Allocated on the heap	(but escape analysis)

Scala and Spire

- `Java primitive types with automatic boxing`
- `BigInt === BigInteger`
- `BigDecimal`

- Scala JS: as JVM with 3 exceptions (https://www.scala-js.org/doc/semantics.html)
- Scala Native: `UInt`, `ULong`, etc… for interoperability

- Spire: `UInt`, `ULong` (using AnyVal, should be opaque types?)
- Spire: Rational, Algebraic, Polynomial, Interval, Complex, Quaternion… (reference types)

On the topic

What Every Computer Scientist Should Know About Floating-Point Arithmetic

https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

How Java's Floating-Point Hurts Everyone Everywhere

https://people.eecs.berkeley.edu/~wkahan/JAVAhurt.pdf

https://github.com/typelevel/spire

https://github.com/denisrosset/minispire

API and laws

https://github.com/typelevel/spire

https://github.com/denisrosset/minispire

Laws

- x <= y and y <= z implies x <= z
  Generally satisfied
  Condition for sorting algorithms
- (x + y) + z = x + (y + z)
  Approximatively satisfied by Double
  Satisfied by BigDecimal
- x * y / y = x
  Approximatively satisfied by Double, BigDecimal
  Satisfied by Rational

(In general: questions of integer overflow, floating-point precision)

Let’s see some code, demos and come back

Questions to audience

- Who has used generic programming? seq.sortBy, seq.sum
  Ordering, Numeric
- Who has used?

  Float, Double
  BigInt
  BigDecimal
  Other types

- Who knew about Spire beforehand?
- Who is using Spire? @denisrosset // physics@denisrosset.com

Spire status

Spire has a very precise core (that includes cats-kernel, algebra)

Too precise? (additive non/commutative semigroups/monoids/groups)

Spire has excellent number types (esp. Rational and Algebraic)

Code quality varies (50% code coverage) but excellent for core

Spire has clever performance hacks (specialization, macros, AnyVal types, etc…)

Scala 3 migration unclear

Spire is ***big*** and is okayish as project health/community goes

Roadmap

- Identify core features with a clear Scala 3 migration path

- Reboot the project *a la* cats (focus on documentation, code coverage, clarity)

- Add features with community traction

- Ask Scala Spree participants
  @denisrosset physics@denisrosset.com // yan9za1@gmail.com
  @bishabosha // philippus@gmail.com // bdx1993@gmail.com