Establishing Orbit with Shapeless

Dave Gurnell

underscore

Get The Book!

http://underscore.io/books/shapeless-guide

(By the way, it’s free)

http://underscore.io/books/shapeless-guide

What is Shapeless?

What is Shapeless?

Library for *generic programming*

Enables new abstractions in Scala

Developed by Miles Sabin + contributors

Why Study Shapeless?

It’s easier than you think

It’ll expand your understanding of Scala

You’re (probably) already using it

Generic Programming

```scala
final case class Employee(
  name        : String,
  number      : Int,
  manager     : Boolean
)

final case class IceCream(
  name        : String,
  numCherries : Int,
  inCone      : Boolean
)
```

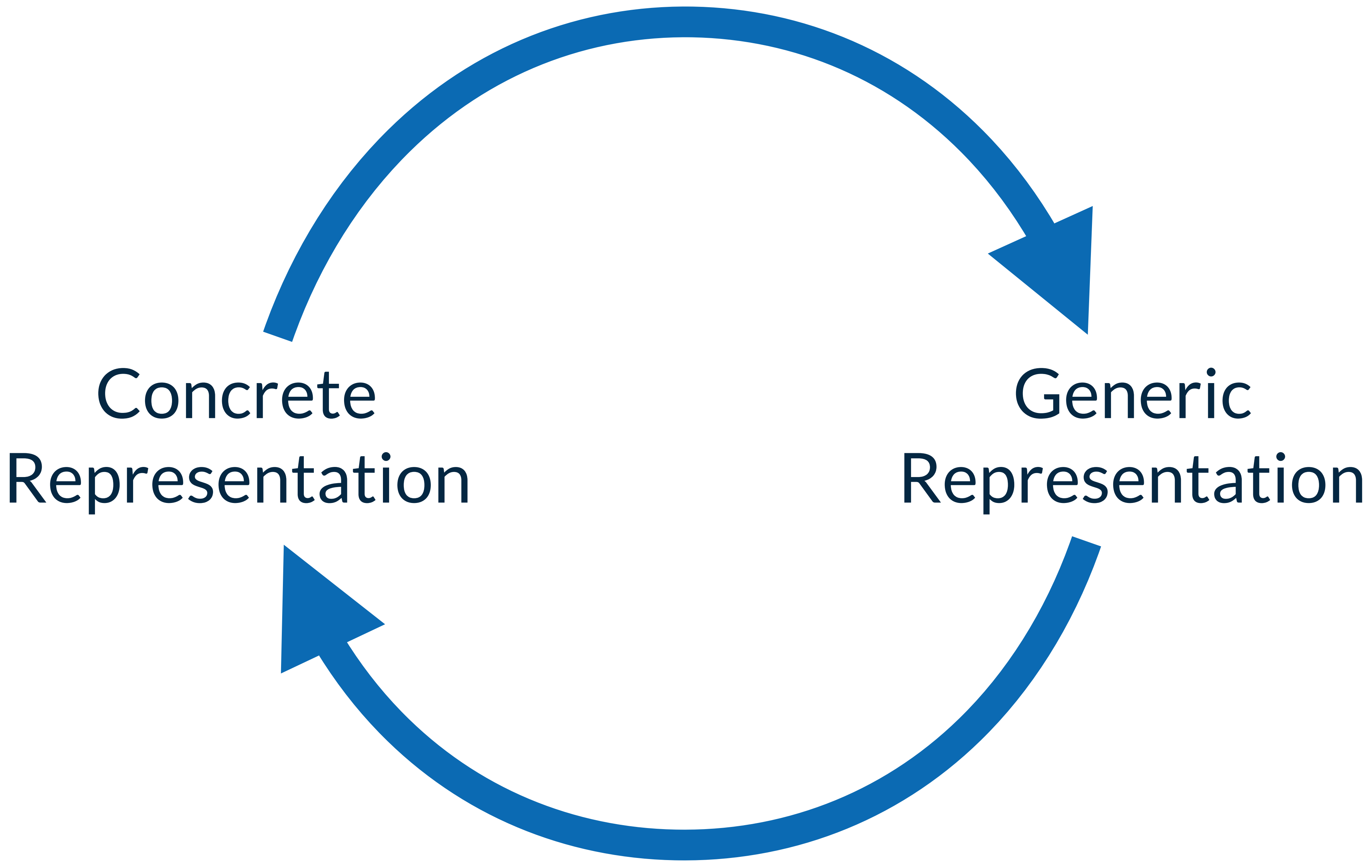
```scala
final case class Employee(
  name        : String,
  number      : Int,
  manager     : Boolean
)

final case class IceCream(
  name        : String,
  numCherries : Int,
  inCone      : Boolean
)
```

```scala
def employeeCsv(e: Employee): List[String] =
  List(
    e.name,
    e.number.toString,
    e.manager.toString
  )

def iceCreamCsv(c: IceCream): List[String] =
  List(
    c.name,
    c.numCherries.toString,
    c.inCone.toString
  )
```

```scala
final case class Employee(
  name       : String,
  number     : Int,
  manager    : Boolean
)

final case class IceCream(
  name       : String,
  numCherries : Int,
  inCone     : Boolean
)
```

```scala
final case class Employee(
  name          : String,
  number        : Int,
  manager       : Boolean
)

final case class Rocket(
  model         : String,
  inSpaaace     : Boolean,
  fuelAmount    : Double
)
```

```scala
final case class Employee(
  name        : String,
  number      : Int,
  manager     : Boolean
)

final case class Dog(
  name        : String,
  breed       : String,
  chasesCars  : Boolean,
  numBoofs    : Int
)
```


Concrete Representation
Generic Representation

Generic Representations

Any

Algebraic Data Type

Products (“and” types)

case classes / case objects

Coproducts (“or” types)

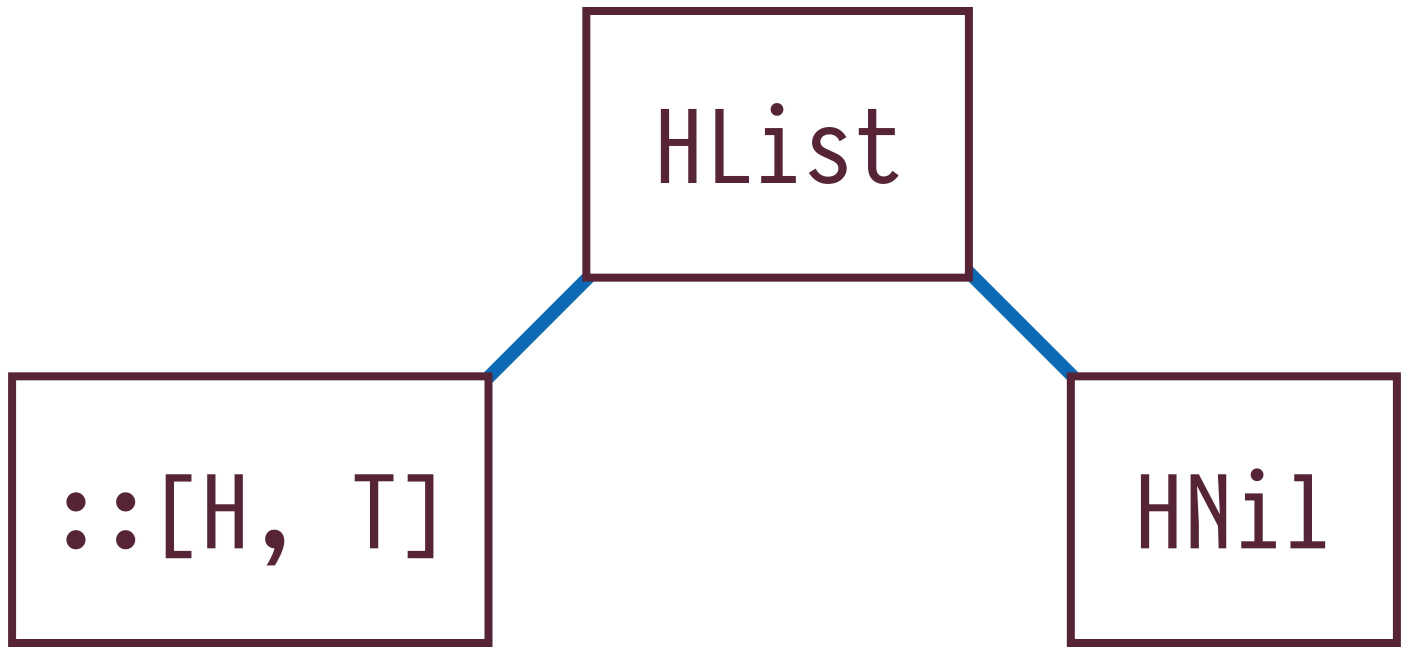
sealed traits / sealed abstract classes

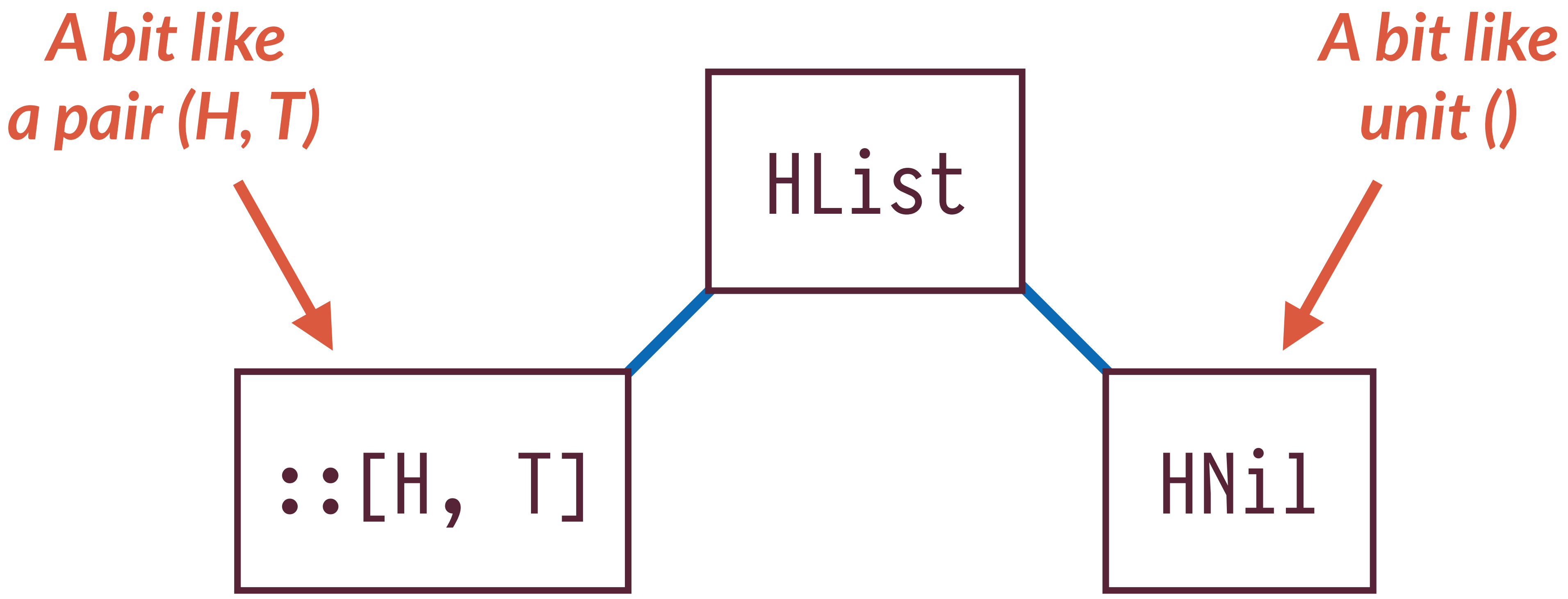
Products (“and” types)

case classes / case objects

Coprodu [illegible] r” types)

sealed traits / se [illegible] bstract classes


HList
::[H, T]
HNil


A bit like
a pair (H, T)
HList
A bit like
unit ()
::[H, T]
HNil

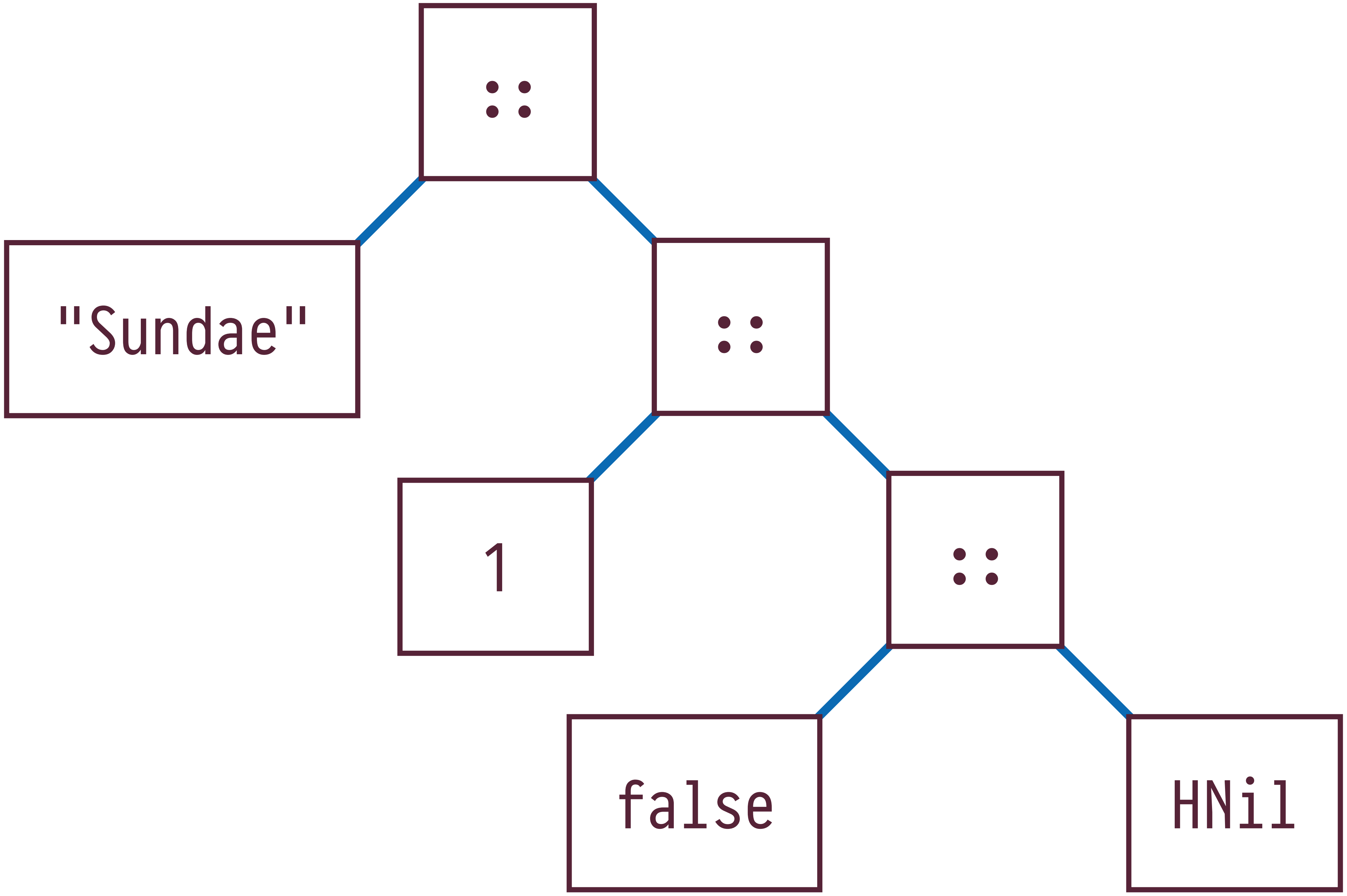
```scala
import shapeless._

type IceCreamRepr =
  ::[String, ::[Int, ::[Boolean, HNil]]]

val iceCream: IceCreamRepr =
  "Sundae" :: 1 :: false :: HNil
```

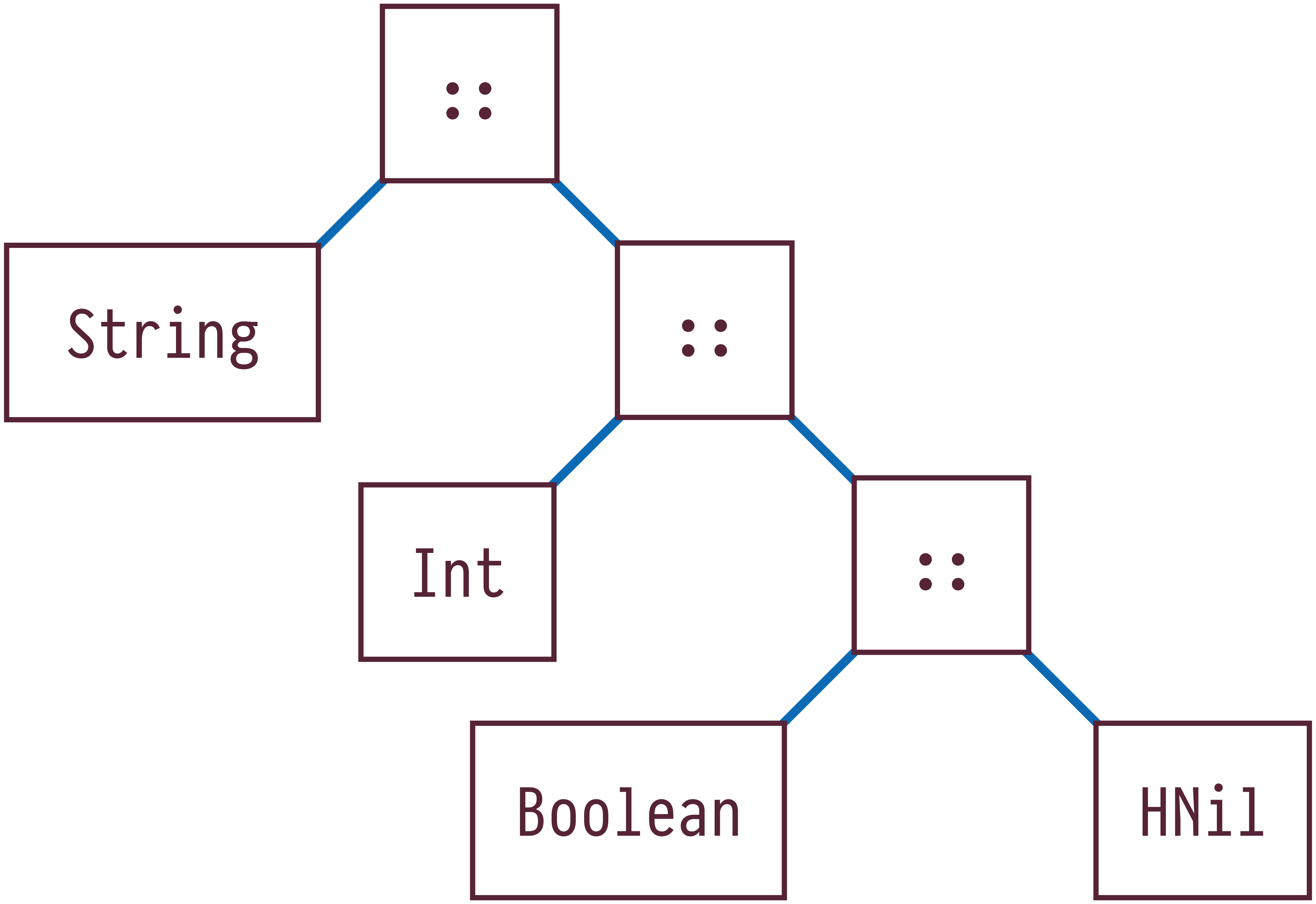
```scala
import shapeless._

type IceCreamRepr =
  String :: Int :: Boolean :: HNil

val iceCream: IceCreamRepr =
  "Sundae" :: 1 :: false :: HNil
```

Demo Time!

representations.scala

Writing Generic Code


::
"Sundae"
::
1
::
false
HNil


::
String
::
Int
::
Boolean
HNil

```
def encodeCsv[A](value: A): List[String] =
  ???
```

Type Classes

```scala
// Type class
trait CsvEncoder[A] {
  def encode(value: A): List[String]
}
```

```scala
// Type class
trait CsvEncoder[A] {
  def encode(value: A): List[String]
}

// Entry point
def encodeCsv[A](value: A)(implicit enc: CsvEncoder[A]) =
  enc.encode(value)
```

```scala
// Type class
trait CsvEncoder[A] {
  def encode(value: A): List[String]
}

// Entry point
def encodeCsv[A](value: A)(implicit enc: CsvEncoder[A]) =
  enc.encode(value)

// Type class instances
implicit val employeeEnc: CsvEncoder[Employee] = ???
implicit val iceCreamEnc: CsvEncoder[IceCream] = ???
```

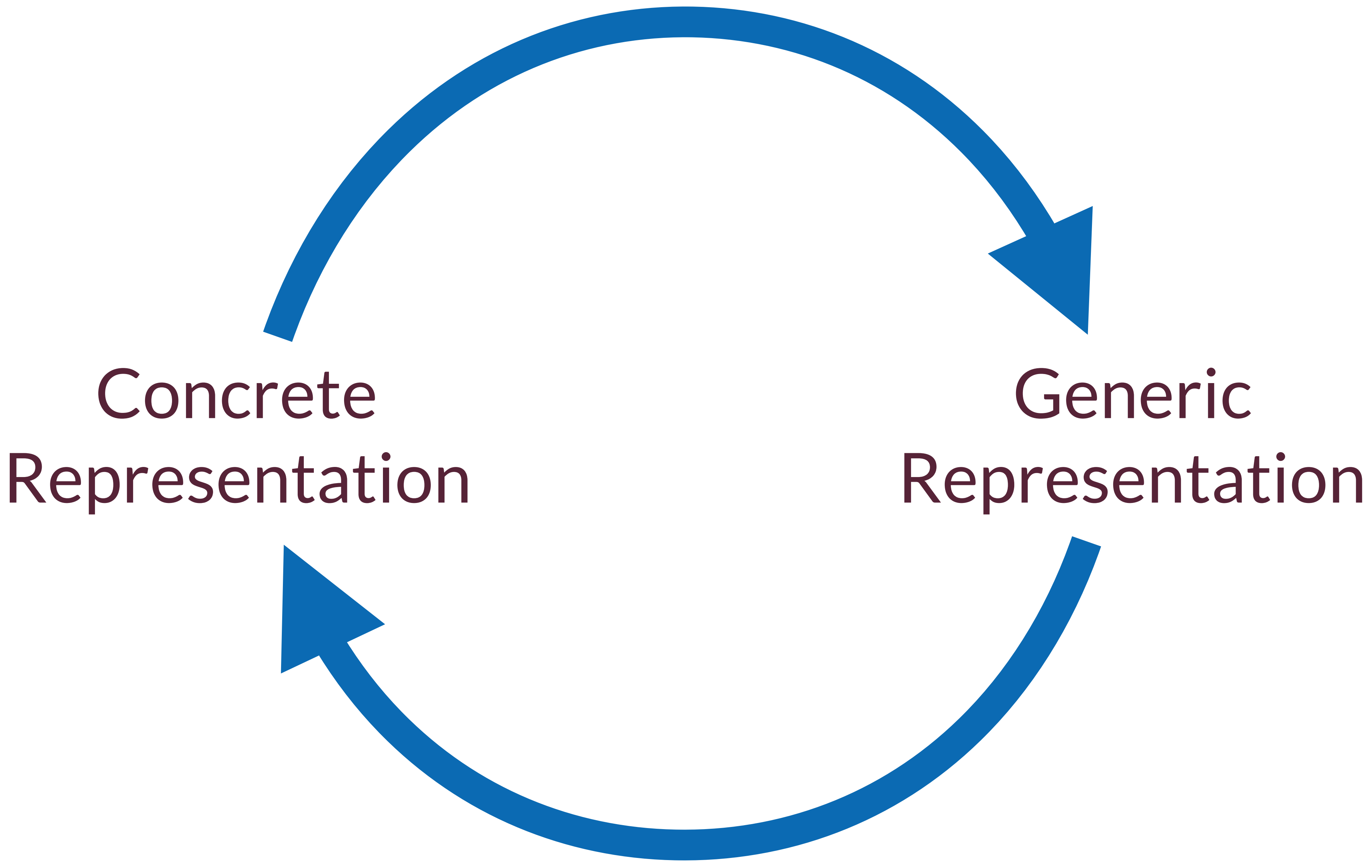
```scala
// Type class
trait CsvEncoder[A] {
  def encode(value: A): List[String]
}

// Entry point
def encodeCsv[A](value: A)(implicit enc: CsvEncoder[A]) =
  enc.encode(value)

// Type class instances
implicit val employeeEnc: CsvEncoder[Employee] = ???
implicit val iceCreamEnc: CsvEncoder[IceCream] = ???

// Use cases
encodeCsv(employee)(employeeEnc)
encodeCsv(iceCream)(iceCreamEnc)
```

```scala
// Type class
trait CsvEncoder[A] {
  def encode(value: A): List[String]
}

// Entry point
def encodeCsv[A](value: A)(implicit enc: CsvEncoder[A]) =
  enc.encode(value)

// Type class instances
implicit val employeeEnc: CsvEncoder[Employee] = ???
implicit val iceCreamEnc: CsvEncoder[IceCream] = ???

// Use cases
encodeCsv(employee)
encodeCsv(iceCream)
```

Demo Time!

csv.scala

Type Class Derivation


Concrete Representation
Generic Representation

```scala
// Empty HList
implicit val hnilEnc: CsvEncoder[HNil] = ???

// Non-Empty HList
implicit def hlistEnc[H, T]: CsvEncoder[H :: T] = ???
```

Demo Time!

csv.scala

Dependent Types

```scala
trait Generic[A] {
  type Repr
  def to(a: A): Repr
  def from(repr: Repr): A
}
```

```scala
def genericify[A](a: A, gen: Generic[A]) =
  gen.to(a)
```

```
def genericify[A](a: A, gen: Generic[A]): gen.Repr =
  gen.to(a)
```

Input type

```
trait Generic[A] {
  type Repr
  def to(a: A): Repr
  def from(repr: Repr): A
}
```

Output type

```scala
trait Generic[A] {
  type Repr
  def to(a: A): Repr
  def from(repr: Repr): A
}

object Generic {
  type Aux[A, R] =
    Generic[A] { type Repr = R }
}
```

```
implicit def genericEnc[A, R](
  implicit
  gen: Generic[A] { type Repr = R },
  enc: CsvEncoder[R]
): CsvEncoder[A] =
  pure(a => enc.encode(gen.to(a)))
```

```
implicit def genericEnc[A, R](
  implicit
  gen: Generic.Aux[A, R],
  enc: CsvEncoder[R]
): CsvEncoder[A] =
  pure(a => enc.encode(gen.to(a)))
```

Demo Time!

csv.scala

Summary

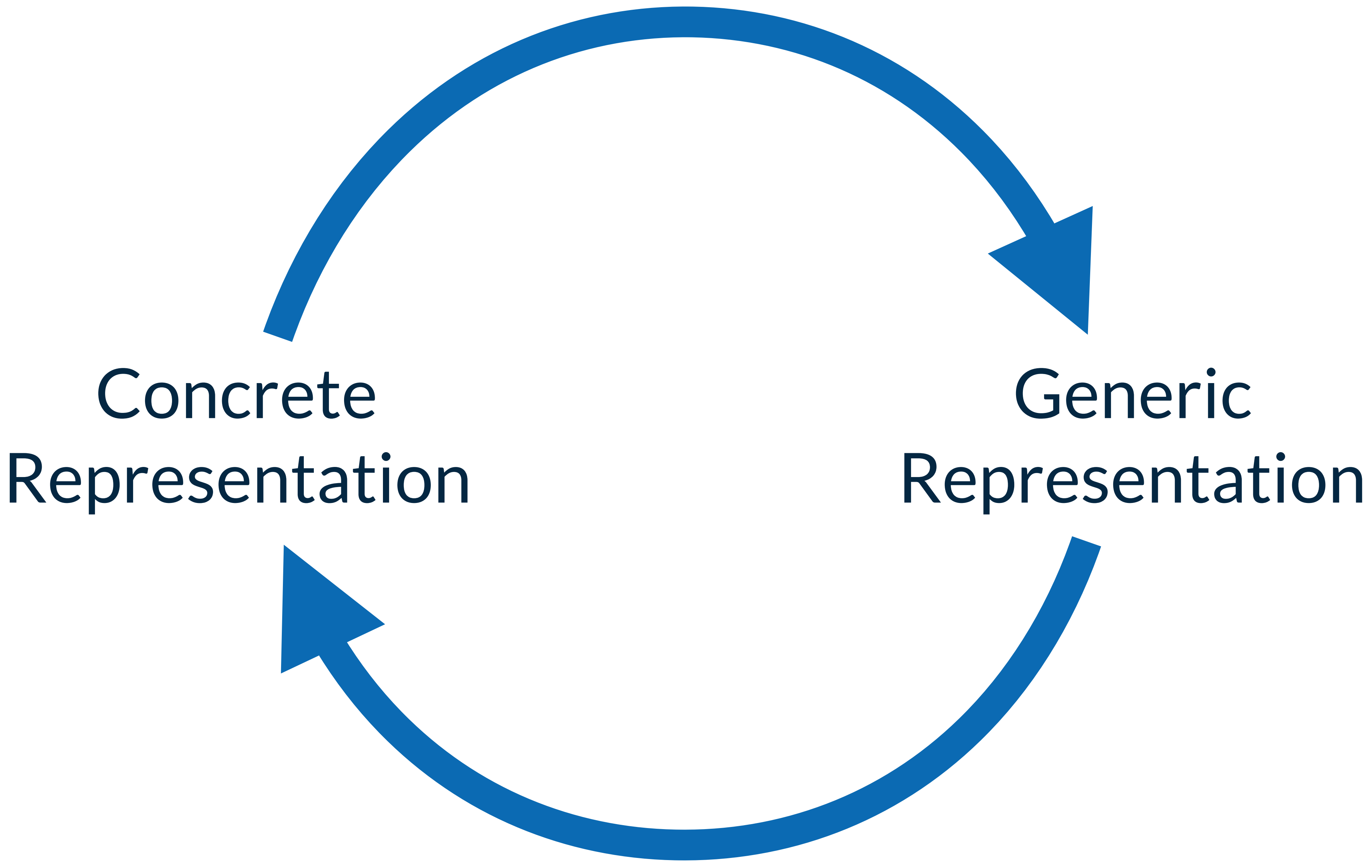
We’ve Covered…

Case classes and generic products (HLists)

The Generic type class

Type class derivation

Dependent types


Concrete Representation
Generic Representation

We’ve Not Covered…

Sealed types and generic coproducts

Implicit divergence and Lazy

Polymorphic functions

Built-in type classes from shapeless.ops

Counting with types

Things We’ve Not Seen…

Instance prioritisation

Performance (cachedImplicit, etc)

Further Reading/Watching

Shapeless for Mortals
Sam Halliday, Scala Exchange 2015

Type Parameters versus Type Members
Jon Pretty, NEScala 2016

The source code for
spray-json-shapeless, argonaut-shapeless,
pureconfig, diff, scalacheck-shapeless

We Like Types!

They prevent mistakes!

They help us write code!

We Like Types!

They prevent mistakes!

They help us write code!

They let the compiler write code for us!

Thanks!

Any Questions?

eBook download
https://underscore.io/books/shapeless-guide

eBook source
https://github.com/underscoreio/shapeless-guide

Example code
https://github.com/underscoreio/shapeless-guide-code

Slides
https://github.com/davegurnell/shapeless-guide-slides