Introduction to programming with dependent types in Scala

Dmytro Mitin

https://stepik.org/2294

April 2017

Values and types

- Types

  Int, Boolean, String, . . .

- Values

  10, **true**, **"abc"**, . . .

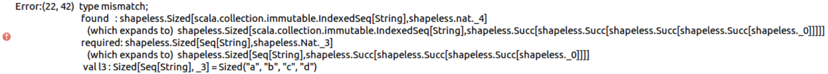
10 : Int
true : Boolean
"abc" : String
. . .

Value functions, type families and dependent types

- Value-level function (value depending on another value)
  (x: Int) => x * x, (x: String) => x + "ab", . . .
- Type family (type depending on another type)
  Seq[Int], Seq[Boolean], Seq[String], . . .
  List[Int], List[Boolean], List[String], . . .
- Dependent type (type depending on value of another type)

Sized[Seq[String], _0], Sized[Seq[String], _1], Sized[Seq[String], _2], . . .

```
val x: Seq[String] = Seq("a", "b", "c")
val x1: Seq[String] = Seq("a", "b", "c", "d")
import shapeless.Sized // Vector type
val x2 : Sized[Seq[String], _3] = Sized("a", "b", "c")
val x3 : Sized[Seq[String], _3] = Sized("a", "b", "c", "d")
```

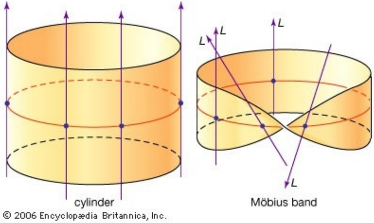


```
Error:(22, 42)  type mismatch;
                found   : shapeless.Sized[scala.collection.immutable.IndexedSeq[String],shapeless.nat._4]
                 (which expands to)  shapeless.Sized[scala.collection.immutable.IndexedSeq[String],shapeless.Succ[shapeless.Succ[shapeless.Succ[shapeless.Succ[shapeless._0]]]]]
                required: shapeless.Sized[Seq[String],shapeless.Nat._3]
                 (which expands to)  shapeless.Sized[Seq[String],shapeless.Succ[shapeless.Succ[shapeless.Succ[shapeless._0]]]]
                val l3 : Sized[Seq[String], _3] = Sized("a", "b", "c", "d")
```

Homotopy type theory (HoTT)

https://homotopytypetheory.org/book/

- Fiber bundle



© 2006 Encyclopædia Britannica, Inc.

- Dependent pair type
  mkPair(*n*, *vn*) !: *Sgma*(*n* !: *Nat*, *Vec*(*n*))
  (_3, *Sized*("a", "b", "c")) : Σ(*n* : *Nat*, Sized[Seq[String], *n*])

Type-level programming

- Types
  True, False, _0, _1, _2, . . . , Int, Boolean, String, . . .
- Values
  True, False, *_0*, *_1*, *_2*, . . .

- Type-level calculations (compile time)
  implicitly[(True# && [False])# || [True] =:= True]
  implicitly[(_1# + [_2])# * [_3] =:= _9]
- Value-level calculations (run time)
  (**true** && **false**) || **true** == **true**
  (True.&&(False)).||(True) == True
  (1 + 2) * 3 == 9
  (*_1*.+(*_2*)).*(*_3*) == *_9*

Theorem proving

```
import provingground._
import HoTT._
import TLImplicits._
import shapeless._
val indN_assoc = NatInd.induc(n :-> (m ~>: (k ~>: (
  add(add(n)(m))(k) =:= add(n)(add(m)(k)) ))))
val hyp = "(n+m)+k=n+(m+k)" :: m ~>: k ~>: (
  add(add(n)(m))(k) =:= add(n)(add(m)(k)) )
val assoc = indN_assoc(m :~> (k :~> add(m)(k).refl))(
  n :~> (hyp :-> (m :~> (k :~>
    IdentityTyp.extnslty(succ)(add(add(n)(m))(k))
      (add(n)(add(m)(k))) (hyp(m)(k))
  ))))
assoc !: n ~>: m ~>: k ~>: (
  add(add(n)(m))(k) =:= add(n)(add(m)(k)) )
```