

# Directembedding

Concealing the Deep Embedding of DSLs

Ólafur Páll Geirsson

École Polytechnique Fédérale de Lausanne  
School of Computer and Communication Sciences



June 11, 2015

# Overview

- 1 Motivation
- 2 Directembedding
- 3 Case study: slick-direct v2
- 4 Conclusion

# Motivation

## Mission statement

Enable wider adoption of embedded DSLs

# The Struggle

Deeply embedded vs. Shallowly embedded

	Author	User
Deep		
Shallow		

# Motivation

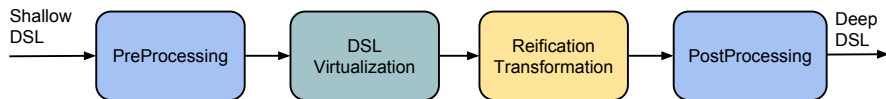
## Problem statement

- Enable DSL authors to write feature rich deeply embedded DSL
- Make it effortless to provide a shallowly embedded interface for DSL users

# Overview

- 1 Motivation
- 2 **Directembedding**
  - Architecture
  - Configuration
  - Language virtualization
  - Type overriding
  - Improved error messages
- 3 Case study: slick-direct v2
- 4 Conclusion

# Pipeline



# Configuration

@reifyAs

```
// Inside Query trait.  
@reifyAs(Take)  
def take(i: Long): Query[T, C] = ???  
// Shallow query.  
query {  
  query.take(1)  
}  
// Deep query.  
Take(lift(query), lift(1))
```



# Configuration

@reifyAsInvoked

```
// Inside Query trait.  
@reifyAsInvoked  
def take(i: Long): Query[T, C] = ???  
// Shallow query.  
query {  
  query.take(1)  
}  
// Deep query.  
lift(query).take(lift(1))
```

# Configuration

@passThrough

```
// Inside Query trait.  
@passThrough  
def take(i: Long): Query[T, C]  
def missingAnnotation(): Int = 1  
// Shallow query.  
query {  
  Query.take(missingAnnotation())  
}  
// Deep query.  
lift(Query).take(missingAnnotation())
```

# Language Virtualization

## Override standard language features

```
if (cond) 1 else 2
// __ifThenElse(cond, 1, 2)
x == y
// infix_==(x, y)
var x = "str"
// __newVar("str")
while (cond) body
// __while(cond, body)
```

# Type overriding

## Override behavior predefined and third-party types

```
class MyInt {  
  @reifyAs(IntPlus)  
  def +(x: Int): Int = ???  
}  
  
// Shallow  
dsl {  
  val x = 1  
  x + 2  
}  
  
// Deep  
IntPlus(x, lift(1))
```

# Improved error messages

```
// Shallow
dsl {
  val s = "foobar"
  s.charAt(1)
}
// Compiler error
[error] method charAt on class String is not
      supported in example.dsl:
s.charAt(1)
  ^
```

# Overview

- 1 Motivation
- 2 Directembedding
- 3 Case study: slick-direct v2**
  - Lifted embedding
  - Direct embedding v1
  - Shadow embedding
  - Direct embedding v2
- 4 Conclusion

# Problem statement

## Problem statement

Develop an awesome embedded query language in Scala

# Lifted embedding

aka Slick

- Currently supported API in Slick 3.0

## The good parts

- 1 Uses standard Scala
- 2 Feature rich

## The bad parts

- 1 Lifted embedding table requires boilerplate
- 2 Cryptic error messages



# Problem 1

Lifted embedding table requires boilerplate

```
case class User(id: Int, name: String)

// Lifted embedding table
class Users(tag: Tag)
  extends Table[User](tag, "User") {

  def id: Rep[Int] = column[Int]("id")
  def name: Rep[String] = column[String]("name")

  def * = ProvenShape.proveShapeOf((id, name) <>
    ((User.apply _).tupled, User.unapply))
}
```

## Problem 2

### Cryptic error messages

```
[Error] User.scala:25: No matching Shape found.  
Slick does not know how to map the given types.  
Possible causes: T in Table[T] does not  
match your * projection. Or you use an  
unsupported type in a Query (e.g. scala List).
```

```
  Required level: slick.lifted.FlatShapeLevel
```

```
    Source type:
```

```
(slick.lifted.Rep[Int], slick.lifted.Rep[String],  
  slick.lifted.Rep[Int])
```

```
  Unpacked type: (Int, String)
```

```
    Packed type: Any
```

```
def * = ProvenShape.proveShapeOf(  
  (id, name, id) <> ((User.apply _).tupled, User.unapply)  
    ^
```

# Direct embedding

Deprecated in Slick 3.0

## The good parts

- 1 No more boilerplate
- 2 Comprehensible error messages

## The bad parts

- 1 Queries can fail at runtime
- 2 Difficult to develop and maintain

# Benefit 1

No more boilerplate

```
@Table("USER")
case class User(
  @PrimaryKey @Column("ID")
  id: Int,
  @Column("NAME")
  name: String
)
```

# Problem 1

Queries can fail at runtime

```
// Compiles!  
Query[User].map(_.id.toDouble)  
// But runtime error...
```

# Shadow embedding

Powered by Yin-Yang

- Master project of Amir Shaikhha, August 2013

## The good parts

- 1 User-friendly API
- 2 Comprehensive and comprehensible error messages
- 3 Great performance

## The not so good parts

- 1 Boilerplate/code duplication

# Shadow embedding

Powered by Yin-Yang

- Master project of Amir Shaikhha, August 2013

## The good parts

- 1 User-friendly API
- 2 Comprehensive and comprehensible error messages
- 3 Great performance

## The not so good parts

- 1 Boilerplate/code duplication

We are almost there!

# Problem

## Auto generated SlickCake

```
trait YYSlickCakeTuples {  
  type Tuple2[T1,T2] = YYTuple2[T1,T2]  
  implicit def yyRepTuple2ToYYTuple[T1,T2](x:  
    CakeRep[scala.Tuple2[T1,T2]]): Tuple2[T1,T2]  
    = x.asInstanceOf[Tuple2[T1,T2]]  
  // 1300 more LOC for up to Tuple22...  
}  
  
trait YYSlickCake with YYSlickCakeTuples {  
  implicit def yyColumnOptionToYYOption[T](x:  
    YYColumn[scala.Option[T]]): YYOption[T]  
    = YYOption.fromYYColumn(x)  
  // ...  
}
```



# Problem

## 3 layers of Query

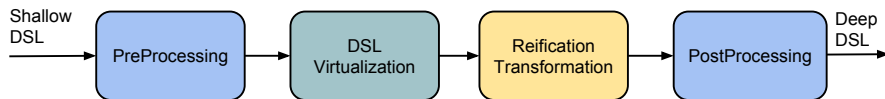
```
trait Query[T] // Shallow
  def map[S](projection: T => S): Query[S]
trait YYQuery[T] // Shadow
  def map[S](projection: YYRep[T] => YYRep[S]):
    YYQuery[S]
    = YYQuery.fromQuery(
      query.map(underlyingProjection(projection))
      (YYShape.ident[S]))
trait Query[E, T] // Lifted
  def map[F, G, T](f: E => F)(implicit shape:
    Shape[F, T, G]): Query[G, T]
```

# Directembedding

From shallow to deep in one step

## slick-direct v2

# Pipeline reminder



# Shallow configuration

## Simplified example

```
trait Query[T, C[_]]
  @reifyAsInvoked
  def filter(f: T => Boolean): Query[T, C]

  @reifyAs(SlickReification.column _)
  def column[T, C](e: T, name: String,
    tt: TypedType[C]): C

  @reifyAs(SlickReification.slick_=== _)
  def infix_==(a: Int, b: Int): Boolean = ???

object SlickPredef
  @passThrough
  def implicitly[T]: T
```

# Deep configuration

## Simplified example

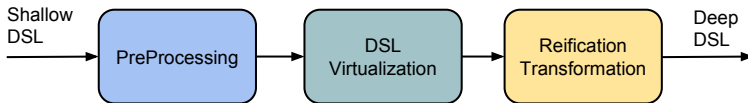
```
object SlickReification {  
  def slick_===[T](lhs: lifted.Rep[T], rhs:  
    lifted.Rep[T]): Rep[Option[Boolean]] = {  
    columnExtensionMethods(lhs) === rhs  
  }  
  def column[T, C](e: AnyRef,  
    field: Rep[String],  
    tt: TypedType[C]): Rep[C]  
  // 20 more LOC  
}
```

# Directembedding transformation

## Pipeline

### Shallow query

```
query {  
  Query[User].filter(_.name == "Olafur")  
}
```

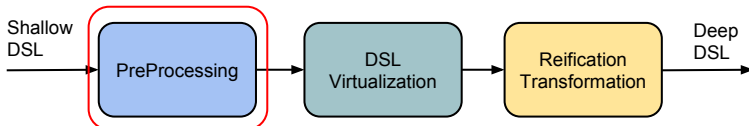


# Directembedding transformation

## Pipeline

### PreProcessing

```
class UserTable extends Table[User] { /* ... */ }  
UserTable.filter { u =>  
  column(u, "name", implicitly[TypedType[String]])  
    == "Olafur"  
}
```

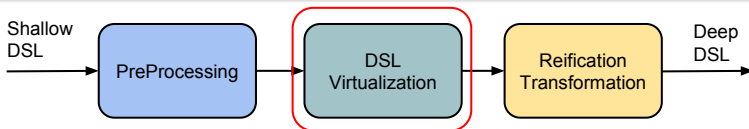


# Directembedding transformation

## Pipeline

### DSL Virtualization

```
import DslConfig._  
class UserTable extends Table[User] { /* ... */  
  UserTable.filter { u => infix_==(  
    shallowColumn(u, "name",  
      implicitly[TypedType[String]]),  
    "Olafur"  
  )  
}
```



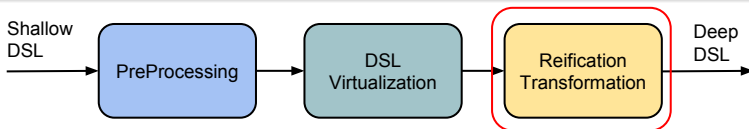


# Directembedding transformation

## Pipeline

### Reification Transformation

```
import DslConfig._  
class UserTable extends Table[User] { /* ... */ }  
lift(UserTable).filter { u => slick_==(  
  deepColumn(u, lift("name"),  
    implicitly[TypedType[String]]),  
  lift("Olafur")  
)  
}
```



# It works!

## FilterSpec

```
"filter" should "work with string equality" in {  
  equalQueries(  
    query {  
      directUsers.filter(_.name == "Olafur")  
    }.result,  
    liftedUsers.filter(_.name === "Olafur").result  
  )  
} // Success
```

# Overview

- 1 Motivation
- 2 Directembedding
- 3 Case study: slick-direct v2
- 4 Conclusion

# Conclusion

## Directembedding

- Language virtualization
- Type overriding
- More reification options
- Improved configuration

## Slick-direct v2

- Under 300 LOC, developed in less than 2 weeks
- Queries supported: select \*, join, filter, map, flatMap, take, (custom column types)

# Future work

- More case studies
- Improve slick-direct

# The End

Thank you!