

Implementing a Universe Type System for Scala

Manfred Stock

ETH Zurich

Master Thesis Final Presentation
11.02.2008

Outline

Architecture

Examples

Conclusion

Introduction

Generic Universe type system

- Support for generics.
- Separation of encapsulation and topology.

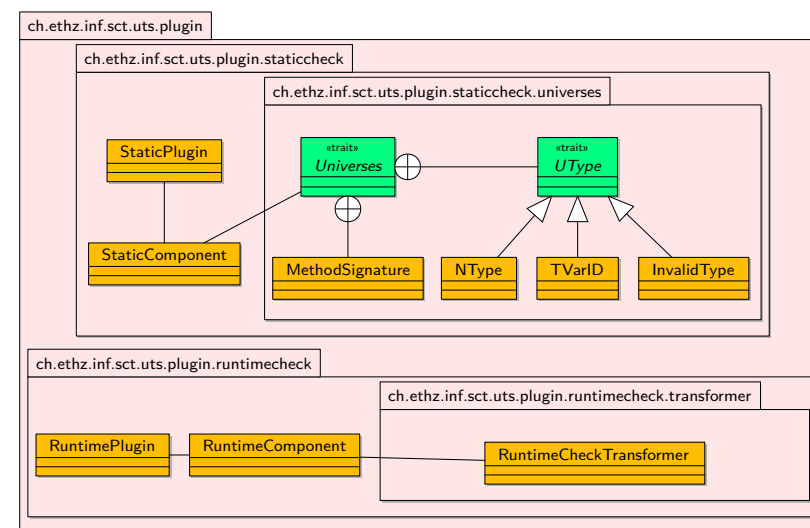
Implementation consists of two compiler plugins

- Static type checks.
 - Simple inference of ownership modifiers.
- Addition of runtime checks.

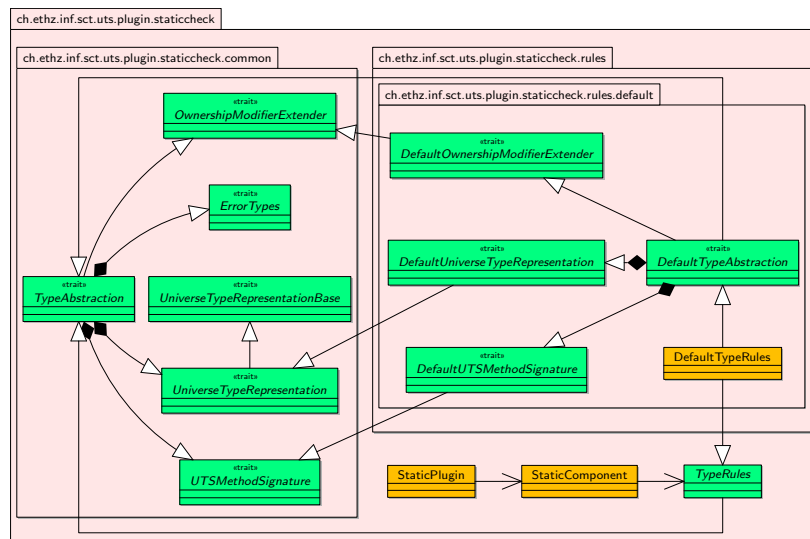
Extensible architecture, one may add new rules easily

- Main change since intermediate presentation.

Architecture - Before



Revamped Architecture - Static Check



Encapsulation vs. Topology

```
package oamexample
import ch.ethz.inf.sct.uts.annotation._

class P {
  5   val a: A @any = new A
      val b         = a.a           // Ok
      a.a           = b             // OaM: Not ok
      a.f           = b             // OaM: Not ok
      a.g           = b             // Ok
  10  a.b           = new Object    // Not ok
}

class A {
  var a: Object @any = _
  var b: Object @peer = _
  15  def f {}
      @pure def g {}
}
```

Generics - A Simple Stack I

```
package stack
import ch.ethz.inf.sct.uts.annotation._

class Stack[T] {
  5   private var stop: Element[T] @rep = null
      def push(value: T) {
          stop = new (Element[T] @rep)(value, stop)
      }
      def pop = {
  10    val res = stop
          if (stop != null) { stop = stop.next }
          res
      }
      @pure def top =
  15    if (stop != null) { stop.value }
          else { null.asInstanceOf[T] }
      @pure def empty = stop == null
}
```

Generics - A Simple Stack II

```
package stack
import ch.ethz.inf.sct.uts.annotation._

class Element[T](val value: T, val next:
  Element[T] @peer)
```

Generics - A Simple Stack III

```

package stack
import scala.testing.SUnit._
import ch.ethz.inf.sct.uts.annotation._

5 object StackTest extends TestConsoleMain {
    def suite = new TestSuite(new STC("Stack"))
    class STC(n: String) extends TestCase(n) {
        def runTest() {
            var s = new (Stack[Int] @rep)
10         s.push(10)
            assertTrue("top failed.", s.top == 10)
            assertTrue("pop failed.", s.pop == 10)
            assertTrue("Not empty.", s.empty)
            println("Done.")
15     }
  }
}

```

Supported Language Subset

Supported features

- ▶ Classes and inheritance.
- ▶ Instance fields, dynamically bound methods, constructors.
- ▶ Operations on objects.
- ▶ Control structures such as *if-then-else* and *match*.

Partially supported features

- ▶ Nested classes and methods.
- ▶ Traits and mixin-composition.
- ▶ Singleton objects.
- ▶ First-class functions.

Runtime Checks

```

package rt
import ch.ethz.inf.sct.uts.annotation._

object RuntimeExample {
5   def main(args : Array[String]) : Unit = {
        val a = new (AnyRef @rep)
        if (a.isInstanceOf[AnyRef @peer]) {
            a.asInstanceOf[AnyRef @peer]
        }
10    else {
        println("Not a peer. But I don't care:")
        a.asInstanceOf[AnyRef @peer]
    }
    println("Done.")
15  }
}

```

Unsupported Language Subset

Unsupported features

- ▶ Self-type annotations.
- ▶ Variance annotations of type parameters.

Inference and Runtime Checks

Inference of ownership modifiers

- ▶ Works for fields, local variables, and method return types.
- ▶ Not implemented yet for type arguments of method calls.

Runtime checks

- ▶ No support for anonymous classes.
- ▶ No support for generics.

Future Work

Basic implementation

- ▶ Complete support of Scala.
- ▶ Recompile Scala class library, add owner field.

Extensions

- ▶ Add runtime support for generics as in [Ott07].
- ▶ Static and runtime inference as in [Lyn05, Kel06].

General

- ▶ Switch to Apache Maven for the build.

Challenges

Finding good abstractions

- ▶ Type representation.

Making architecture extensible

- ▶ Access to inner classes.
- ▶ Separating traversal from checks.

References



Nathalie Kellenberger.

Static Universe Type Inference.

Master's thesis, Swiss Federal Institute of Technology Zurich (ETHZ), Department of Computer Science, 2005–2006.



Frank Lyner.

Runtime Universe Type Inference.

Master's thesis, Swiss Federal Institute of Technology Zurich (ETHZ), Department of Computer Science, 2005.



Mathias Ottiger.

Runtime Support for Generics and Transfer in Universe Types.

Master's thesis, Swiss Federal Institute of Technology Zurich (ETHZ), Department of Computer Science, 2007.

Revamped Architecture - Runtime Check Addition

ch.ethz.inf.sct.uts.plugin.runtimecheck

