Project Kepler

Eugene Burmako

EPFL, LAMP

03 July 2012

Previously on Kepler

- metaprogramming is useful
- macros = compile-time AST transformers
- prototype of macros for Scala

Nowadays

- there is a spec for macros
- this spec is provisionally accepted for inclusion in Scala
- starting from 2.10.0-M3 Scala has macros
- the road to production has led to a number of discoveries

Evolution

v1:

```
macro def assert(cond: Boolean, msg: Any) =
  if (assertionsEnabled)
    c"if (!$cond) raise(msg)"
  else
    c"()"
```

v2:

```
def assert(cond: Boolean, msg: Any) = macro assertImpl
def assertImpl
    (c: Context)
    (cond: c.Expr[Boolean], msg: c.Expr[Any]) =
  if (assertionsEnabled)
    c.reify(if (!cond.splice) raise(msg.splice))
  else
    c.reify(())
```

Reify

The main insight we gained.

Kills two birds with one stone:

- implements a quasiquoting facility
- achieves hygiene

Hygiene

Macro expansions are inlined into the call site.

What happens to bindings?

```
object Assert {
  def raise(msg: Any) = throw new AssertionError(msg)
  macro def assert(cond: Boolean, msg: Any) =
    <[ if (!cond) raise(msg)) ]>
}

object Test extends App {
  def raise(msg: Any) = { /* haha, tricked you */ }
  assert(2 + 2 == 3, "no way")
}
```

Achieving hygiene

In Common Lisp they have gensym.

In Scheme and Nemerle they perform transparent alpha-renaming.

In Scala we don't do any of this.

A fascinating discovery

Non-hygienic macros can be bootstrapped into hygienic macros (in that sense, our macros are self-cleaning).

Apparently that's old news

Macros as Multi-Stage Computations Ganz, Sabry & Taha

Staged Notational Definitions Taha & Johann

Yet it's a novel interpretation

Taha et al. build a macro system atop of a staged language.

We build a staged system atop of a macro language.

Deliverables

- Paper: Scala Macros, a Technical Report (META'2012)
- Documentation: http://scalamacros.org
- Code: http://github.com/scala/scala

Behind the scenes

- polymorphic macros and interaction with type inference
- reify defeats erasure
- ultra-cake pattern in the new Scala reflection

Future work

- Marriage of macros and type inference
- Better reify
- Macro types

Thanks!

eugene.burmako@epfl.ch