Project Kepler

Eugene Burmako

EPFL, LAMP

17 January 2012

What?

My research is about compile-time metaprogramming, i.e. about answering the question: “how to empower developers so that they can extend the compiler, but stay sane and not mess it up?”

Why?

CTM enables the following techniques:

- language virtualization
- program reification
- self-optimization
- algorithmic program construction

How?

Macros and quasiquotes. The former make extending the compiler possible, the latter make it bearable.

But!

Q: Scala has enough advanced features for its creator to think about introducing feature switches. Why bother?

A: CTM lets us advance in several areas that are hot for the community: code lifting for better DSLs, domain-specific optimization for high performance, (speculation) type-level computations for principled type hackery.

Zen

```
class Queryable[T, Repr](query: Query) {
  macro def filter(p: T => Boolean): Repr = scala"""
    val b = $newBuilder
    b.query = Filter($query, ${reify(p)})
    b.result
  """
}
```

Now

Prototypes: http://github.com/scalamacros/kepler (macro defs, quasiquotes, splicing, pattern matching)

Documentation: http://scalamacros.org (use cases, talks and walkthroughs, alpha versions of proposals)

Use

- Slick language integrated connector kit
- Lenses
- Shapeless
- Domain-specific inlining

Next?

SIP, stabilization, macro types and macro annotations.

Thanks!

eugene.burmako@epfl.ch