Hello, scalac

Adriaan Moors, EPFL

Scalathon, 16 July 2011

Pick your battles

- Scalac is big, but modular.
- Can be productive without knowing all of it (I kind of am even though I don't).
- Let's get you started.

Start small

- Fix a bug with a clear solution.
  - Take, take -- we have plenty!
- Write a plugin
  - http://lmgtfy.com/?q=scalac+plugin+tutorial

Diagnosis

- **-Xprompt** will give you a stack trace for an error (warnings can be made fatal too)
- **-Xprint:typer** prints the program after type checking (incl. implicits and inference)
- **-Ytyper-debug**, **-Ydebug** produce debug output (a lot of it -- see -Xprompt)

Minimizing the test case

- Do you really need higher-kinded types to trigger the problem?
- Have distinct names for everything (if synonyms are essential, use -uniqid).
- Minimize, vary, minimize, vary, minimize.

Your own build

- **ant locker.done** uses the stable reference compiler to build the local reference compiler
- **ant quick.bin** uses locker to build the compiler you'll be fiddling with (in build/quick/bin)
- when you're done, **ant clean test**

Debugging

- Eclipse and IntelliJ both work well enough.
- I still use TextMate and lots and lots of printlns.

Data Structures

- **Symbol**
  - result of resolving an identifier (done lazily in Namers when info is called)
  - sym.info: type that describes the symbol, evolves over time (=phases); time travel using atPhase { ... }
    - timid little brother: rawInfo

Data Structures

- **Type**
  - ClassInfoType // info for a class
  - MethodType // for one arglist of a method
  - TypeBound(T, U) // >:T <: U (for abs type)
  - TypeRef(pre, sym, args) // pre.sym[args]
  - PolyType(tps, tp) // type function

Data Structures

- **Tree** abstractly represents the program text
- type checker attributes them with their types (mutable)
- phases traverse and transform (immut)

Phases

- parser, namer, packageobjects: load symbols
- typer: attribute trees with types
- superaccessors: super calls in traits, etc.
- pickler: serialize symbol table
- refchecks: check overriding, type apps

Phases

- selectiveanf, liftcode, selectivecps
- uncurry: FP -> OO
- tailcalls, specialize, explicitouter
- erasure: Scala -> JVM types

Phases

- lazyvals, lambdalift, constructors
- flatten, mixin, cleanup
- icode, <optimiser>, jvm

Operations: TypeMap

```
trait SubstMap[T](from: List[Symbol], to: List[T]) extends TypeMap {
  def apply(tp: Type): Type = if (from.isEmpty) tp else {
    mapOver(tp) match {
      case TypeRef(NoPrefix, sym, args) =>
        appliedType(subst(tp, sym, from, to), args)
      case SingleType(NoPrefix, sym) =>
        subst(tp, sym, from, to)
      case _ =>
        tp
    }
  }
}
```

Operations

LIVE DEMO

```
Last login: Fri Jul 15 18:46:21 on ttys006
dragonfly:~ adriaan$ scal
-bash: scal: command not found
```

```
Last login: Fri Jul 15 18:46:21 on ttys006
dragonfly:~ adriaan$ scal
-bash: scal: command not found

dragonfly:~ adriaan$ scala

Welcome to Scala version 2.10.0.r25281-b20110713122956 (Java HotSpot(T
64-Bit Server VM, Java 1.6.0_26).
Type in expressions to have them evaluated.
Type :help for more information.
```

```
Last login: Fri Jul 15 18:46:21 on ttys006
dragonfly:~ adriaan$ scal
-bash: scal: command not found

dragonfly:~ adriaan$ scala

Welcome to Scala version 2.10.0.r25281-b20110713122956 (Java HotSpot(T
64-Bit Server VM, Java 1.6.0_26).
Type in expressions to have them evaluated.
Type :help for more information.
```

```
scala> :power
** Power User mode enabled - BEEP BOOP SPIZ **
** :phase has been set to 'typer'.          **
** scala.tools.nsc._ has been imported      **
** global._ and definitions._ also imported **
** Try  :help,  vals.<tab>,  power.<tab>    **
```

```
Last login: Fri Jul 15 18:46:21 on ttys006
dragonfly:~ adriaan$ scal
-bash: scal: command not found

dragonfly:~ adriaan$ scala

Welcome to Scala version 2.10.0.r25281-b20110713122956 (Java HotSpot(T
64-Bit Server VM, Java 1.6.0_26).
Type in expressions to have them evaluated.
Type :help for more information.
```

```
scala> :power
** Power User mode enabled - BEEP BOOP SPIZ **
** :phase has been set to 'typer'.           **
** scala.tools.nsc._ has been imported       **
** global._ and definitions._ also imported **
** Try  :help,  vals.<tab>,  power.<tab>     **

scala> import intp.global._
import intp.global._
```

```
scala> abstract class D[T] { def m: T }
defined class D
```

```
scala> abstract class D[T] { def m: T }
defined class D

scala> abstract class C extends D[Int]
defined class C

scala>
```

```
scala> abstract class D[T] { def m: T }
defined class D

scala> abstract class C extends D[Int]
defined class C

scala>

scala> val symD = intp("D")
symD: $r.intp.global.Symbol = class D
```

```
scala> abstract class D[T] { def m: T }
defined class D

scala> abstract class C extends D[Int]
defined class C

scala>

scala> val symD = intp("D")
symD: $r.intp.global.Symbol = class D

scala> val symC =  intp("C")
symC: $r.intp.global.Symbol = class C
```

```
scala> val trT = TypeRef(NoPrefix,
                         symD.typeParams(0), Nil)
trT: Type = T
```

```
scala> val trT = TypeRef(NoPrefix,
                         symD.typeParams(0), Nil)
trT: Type = T

scala> trT.asSeenFrom(ThisType(symC), symD)
res0: Type = Int
```

```
scala> val trT = TypeRef(NoPrefix,
                         symD.typeParams(0), Nil)
trT: Type = T

scala> trT.asSeenFrom(ThisType(symC), symD)
res0: Type = Int

scala> val args = symD.typeParams map
                    ThisType(symC).memberType

args: List[Type] = List(Int)
```

```
scala> val cBTd = symC.tpe.baseType(symD)
cBTd: Type = D[Int]
```

```
scala> val cBTd = symC.tpe.baseType(symD)
cBTd: Type = D[Int]

scala> val cBTS = symC.tpe.baseTypeSeq
cBTS: BaseTypeSeq
  = BTS(C, D[Int], ScalaObject, Object, Any)
```

Questions! Thank you!