Adding support for CLR Generics

Miguel Garcia
http://lamp.epfl.ch/~magarcia

LAMP, EPFL

2011-09-06

Outline

Status

- The original plan: Two easy steps
- What is "Partial Erasure"

Next steps

- What's missing 1 of 4: "Type var bridges for inner classes"
- What's missing 2 of 4: Changes in `AddInterfaces`
- What's missing 3 of 4: "Type var bridges for local defs"
- What's missing 4 of 4: "Type var bridges for abstract type bindings"

Interplay Generics-Statics (`mixin`, `cleanup`)

- Background
- `cleanup` and non-fixed formals

Ideas for further work

Outline

Status
- The original plan: Two easy steps
- What is “Partial Erasure”

Next steps
- What’s missing 1 of 4: “Type var bridges for inner classes”
- What’s missing 2 of 4: Changes in `AddInterfaces`
- What’s missing 3 of 4: “Type var bridges for local defs”
- What’s missing 4 of 4: “Type var bridges for abstract type bindings”

Interplay Generics-Statics (`mixin`, `cleanup`)
- Background
- `cleanup` and non-fixed formals

Ideas for further work

Step 1: Breaking up `AddInterfaces` and `Erasure`:

```
...
explicitouter
"addifaces"
lazyvals
"full-erasure" (without AddInterfaces)
lambdalift
...
```

Step 2: Add "`partial-erasure`", do "`full-erasure`" last:

```
...
explicitouter
"addifaces"
lazyvals
"partial-erasure"
lambdalift
...
cleanup
"full-erasure"
...
```

The p-erasure `|T|` of a type `T` is:

1. For a constant type, itself. For every other singleton type, the p-erasure of its supertype.
2. For other (non-array) typerefs, as follows. When the typeref is to:
   - 2.1 `Any`, `AnyVal`, `scala.Singleton`, or `scala.NotNull`, its p-erasure is `AnyRef`.
   - 2.2 `Unit`, its p-erasure is `scala.runtime.BoxedUnit`.
   - 2.3 `P.C[Ts]` where `C` refers to a class, its p-erasure is `|P|.C`. (where `P` is first rebound, see ticket 2585)
   - 2.4 a non-empty type intersection (possibly with refinement): the p-erasure of its intersection dominator (Scala) or of its first parent (Java)
   - 2.5 `else apply(sym.info) // alias type or abstract type`
3. For "quantified types" (polymorphic or existential), the p-erasure of its result type.
4. For method types:
   - 4.1 For a method type `(Fs)scala.Unit`, `(|Fs|)scala#Unit`
   - 4.2 For any other method type `(Fs)T`, `(|Fs|)|T|`.
5. For a type intersection (possibly with refinement)
   - 5.1 Non-empty: In Scala, the p-erasure of the intersection dominator. In Java, the p-erasure of the first parent.
   - 5.2 Empty: `java.lang.Object` (because the intersection dominator of `Nil` is `AnyRef`)
6. For an annotated type, the p-erasure of its underlying type (where underlying is the type without the annotation)
7. For the classinfo type of
   - 7.1 `java.lang.Object` or a Scala value class, the same type without any parents.
   - 7.2 `Array`, the same type with only `AnyRef` as parent.
   - 7.3 any other classinfo type with parents `Ps`, the same type with parents `|Ps|`, without duplicate `Object` refs.
8. for all other types, the type itself (with any sub-components erased)

Actually only the following differs from full-erasure:

- in the `TypeMap`:
  - For array typerefs, `Array[|T|]`.
- in `transformInfo()`:
  - a type var gets a `TypeBounds info`, with upper bound partially erased and `Nothing` as lower bound.

As we'll see next, partial erasure is necessary but not sufficient ...

Outline

Status
- The original plan: Two easy steps
- What is “Partial Erasure”

Next steps
- What’s missing 1 of 4: “Type var bridges for inner classes”
- What’s missing 2 of 4: Changes in `AddInterfaces`
- What’s missing 3 of 4: “Type var bridges for local defs”
- What’s missing 4 of 4: “Type var bridges for abstract type bindings”

Interplay Generics-Statics (`mixin`, `cleanup`)
- Background
- `cleanup` and non-fixed formals

Ideas for further work

In CLR, "*type params aren't visible in nested types*".

```
class O[T] {
  class I { def f(): T = f(); def m(arg: T) {} }
  def g(i: I): T = i.f()
}
```

- BTW, the reference to `T` inside `I` has no prefix.
- Solution: Add "bridging type vars". A dedicated phase right before `explicitouter` seems advantageous (tentative name: "`tvarbridges4inner`")

```
class O[T] {
  class I[U] { def f(): U = f(); def m(arg: U) {} }
  def g(i: I[T]): T = i.f()
}
```

```
trait Lst[+T] {
  def append[U >: T](other: U): Lst[U] = append(this)
  def append[U](other: Lst[U]): Lst[U] = other
}
class CLst[T] extends Lst[T]
```

Changes needed in `AddInterfaces` (specifically, `LazyImplClassType`) because ...

```
[[syntax trees at end of addifaces]]
  . . .
  /*- PROBLEM: dangling T */
  abstract trait Lst$class extends java.lang.Object
  with ScalaObject with Lst[T] {

    def /*Lst$class*/$init$(): Unit = { () };

    def append[U >: T <: Any](other: U): Lst[U] =
      Lst$class.this.append[T](Lst$class.this);

    def append[U >: Nothing <: Any](other: Lst[U]): Lst[U] = other
  }
```

A local def uses a non-local type param

```
object Obj {
  def ownsTypeParamAndLocalClass[T](t: T): T = {

    class LC { def lcm(lcmarg: T): T = lcmarg }

    (new LC).lcm(t)
  }
}
```

A new phase right before `lambdalift` (tentative name: `typevarbridges4local`):

```
object Obj {
  def ownsTypeParamAndLocalClass[T](t: T): T = {

    class LC[U /*- bridge to T */ ] { def lcm(lcmarg: U): U = lcmarg }

    (new LC[T]).lcm(t)
  }
}
```

```
abstract class C { type T; def t(): T; }

object Obj2 {
  def f(x: C): String = {
    class D { def m(t: x.T): String = t.toString(); }

    (new D).m(x.t()) /*- path-dependence allows concluding that
the actual arg to m's invocation conforms to the param's declared type. */
  }
}
```

Another phase, this time before partial erasure, transforming as follows:

```
abstract class C[T] { def t(): T; }

object Obj2 {
  def f[T](x: C[T]): String = {
    class D { def m(t: T): String = t.toString(); }

    (new D).m(x.t()) /*- No path-dependence here. */
  }
}
```

Separate compilation? (Say, `Obj2.f()` not accessing `C#T`)

Outline

- Status
  - The original plan: Two easy steps
  - What is "Partial Erasure"
- Next steps
  - What's missing 1 of 4: "Type var bridges for inner classes"
  - What's missing 2 of 4: Changes in `AddInterfaces`
  - What's missing 3 of 4: "Type var bridges for local defs"
  - What's missing 4 of 4: "Type var bridges for abstract type bindings"
- Interplay Generics-Statics (`mixin`, `cleanup`)
  - Background
  - `cleanup` and non-fixed formals
- Ideas for further work

Statics are per-type-instantiation on CLR. From C# 2.0 spec:

> *A static variable in a generic class declaration is shared amongst all instances of the same closed constructed type, but is not shared amongst instances of different closed constructed types . . . regardless of whether the type of the static variable involves any type parameters or not.*

The CLR way: class-level type-params are visible in static members. For example, the following C# program prints `0050`:

```
class Gen<T> { public static int X = 0; }
class Test {
 static void Main() {
  Console.Write(Gen<int>.X); Console.Write(Gen<string>.X);
  Gen<int>.X = 5;
  Console.Write(Gen<int>.X); Console.Write(Gen<string>.X);
 }
}
```

1. Consequences for `mixin`: TODO
2. More on CLR Generics:

   `http://lamp.epfl.ch/~magarcia/ScalaNET/slides/TourCLRGenerics.pdf`

```
def gy[Y] (y: Y, x : { def f[T](a: T): Int }) = x.f(y)

val ostr = new { def f(a: String) = 4 }
val oint = new { def f(a: Int) = 4 }
val oobj = new { def f(a: Object) = 4 }
val ogen = new { def f[T](a: T) = 4 }
```

If T binds to a concrete type at a callsite, we have fixed-types for formals. However, T can also bind to another type var (Y in the example). Looks like that should be rejected. Some cases:

```
error: type mismatch;
 found : Test.oint.type (with underlying type java.lang.Object{def f(a: Int): Int})
 required: AnyRef{def f[T](a: T): Int}
   gy(123, oint)
           ^      /*- similarly for ostr and oobj. */

 gy(null, null) /*- accepted, NullPointerException at runtime. */
 gy(null, ogen) /*- runs ok. */
 gy(null, oobj.asInstanceOf[ AnyRef{ def f[T](a: T): Int } ]) /*- runs ok too. */
```

Details on how to avoid cache fragmentation at

http://lamp.epfl.ch/~magarcia/ScalaCompilerCornerReloaded/2011Q3/cleanup2.pdf

Outline

- Status
  - The original plan: Two easy steps
  - What is "Partial Erasure"
- Next steps
  - What's missing 1 of 4: "Type var bridges for inner classes"
  - What's missing 2 of 4: Changes in `AddInterfaces`
  - What's missing 3 of 4: "Type var bridges for local defs"
  - What's missing 4 of 4: "Type var bridges for abstract type bindings"
- Interplay Generics-Statics (`mixin`, `cleanup`)
  - Background
  - `cleanup` and non-fixed formals
- Ideas for further work

Once *partially erased* types are available,

1. types can be checked for CLR suitability right after `lambdalift` (they won't get any simpler afterwards)
2. *perhaps* `specialize` has an easier time running later in the pipeline (thus handling simpler AST shapes)
3. program verification tools can get more precise information all the way to ICode.