# Playground for LINQ Expression Trees

© Miguel Garcia, LAMP,
École Polytechnique Fédérale de Lausanne (EPFL)
`http://lamp.epfl.ch/~magarcia`

July 10th, 2010

**Abstract**

In Scala.NET, `TypeParser` enters generics-aware type symbols. These notes show what's been done and what's still missing to support LINQ in Scala.NET. Two caveats: CIL is not being emitted for generics yet, nor is LINQ textual syntax desugared to its SQO formulation.

DRAFT

## Contents

## 1 Playground for generic APIs

Typechecking Scala.NET programs involving generics works as reported in §3.3 and §3.4 of:

- *Doing for assemblies what `ClassfileParser.sigToType()` does for classfiles* `http://lamp.epfl.ch/~magarcia/ScalaCompilerCornerReloaded/SigToType.pdf`

Regarding notation, a minor workaround is (as of now) needed. At the level of assembly metada, `System.Collections.Generic.List` is not called like that but:

```
System.Collections.Generic.List`1
```

(I don't mean you're supposed to write programs that way, I'm just reporting a temporary fix for an audience of compiler hackers, agreed?). One may save typing by having a header with type alias definitions of the form:

```
type List[T] = System.Collections.Generic.‘List\1401‘[T]
```

`‘List\1401‘` is also part of the workaround, as described in the notes. On the bright side, the names of generic methods don't have grave accents, and moreover the notational burden can be hidden using type aliases as shown above.

# 2 Details about SQO-desugared ASTs

These notes build upon:

- *Adding LINQ-awareness to Scala.NET* `http://www.sts.tu-harburg.de/people/mi.garcia/ScalaCompilerCorner/ScalaNetLearnsLINQTricks.pdf`
- *Translation of LINQ queries*, Kaichuan Wen [1, Ch. 4]

## 2.1 Desugaring performed by the C# compiler

The LINQ queries shown below are desugared (and typechecked and etc.) into what Listing 1 shows.

```
private static void Test4()
{
    int[] primes = { 1, 3, 13 }; // , 17, 23, 5, 7, 11
    IQueryable<int> smallPrimes = from q in primes.AsQueryable()
                                  where q < 11
                                  select q;

    System.Linq.Expressions.Expression<Func<int, decimal?>>
      calcPrice = (p) => p * 1.19m;
}
```

The relevant classes are:

- `System.Linq.Expressions.Expression`
- `System.Linq.Expressions.ParameterExpression`
- `System.Reflection.MethodInfo`

For this particular example, the following method also is invoked in the desugared version:

```
public static implicit operator decimal(int value);
Declaring Type: System.Decimal
Assembly: mscorlib, Version=4.0.0.0
```

Finally, CIL details in their full glory for the C# code from Listing 1 appear in Listing 4 on p. 10 and in Listing 5.

For comparison Listing 6 shows the compiled CIL for the already-desugared (SQO formulation) of the first query, i.e.

DRAFT

```
IQueryable<int> smallPrimes = primes.AsQueryable().Where(q => q < 11).Select(q => q);
```

## 2.2 Non-verbose SQO thanks to Scala's type inference

The code in Listing 1 is only an approximation to what a Scala.NET developer would write. With Scala, the code is less verbose (Listing 2) because of type inference.

> TODO finish translating from C# style to Scala.NET style (e.g., express `(MethodInfo) methodof(decimal.op_Implicit)` in Scala syntax (using `System.Reflection`) )

From the point of view of the compiler plugin automating such desugaring, Scala's type inference is also a boon because the transformation has to create an AST with less nodes.

## 2.3 Factory methods for Expression Trees

Each SQO operator has as counterpart one or more factory methods in `System.Linq.Queryable` (a concrete static class). Its methods are in most cases polymorphic (on the element type of the resulting `IQueryable`). In a few cases, there's an overloaded non-generic version. For example:

- `Average(IQueryable<Decimal>)`
- but not in `Count<TSource>(IQueryable<TSource>)`

Details at http://msdn.microsoft.com/en-us/library/system.linq.queryable_members(v=VS.100).aspx

The other type with static factory methods for Expression trees is the abstract class `System.Linq.Expressions.Expression`. It also defines factory methods, which are in all cases non-polymorphic (granted, sometimes the type of one of its parameters is generic).

As shown in Listing 4, the shorthand notation for "extension methods" is a compiler fiction. Instead, by the end of the compilation pipeline, the factory methods in `Queryable` are invoked as static methods. For example:

```
// Scala code
val i32 = Type.GetType("System.Int32")
val msArray = System.Array.CreateInstance(i32, 3)
val q = System.Linq.Queryable.AsQueryable(msArray)
val q2 = System.Linq.Queryable.Where(q1, ...)
```

> TODO Add support for "extension methods", following the recipe at §1 in `http://lamp.epfl.ch/~magarcia/ScalaCompilerCornerReloaded/CILMdAsScala.pdf`

## 2.4 Type params in the method signatures of a `MulticastDelegate` subclass

There are overloads for the `Lambda` factory method (Figure 1 on p. 13) distinguishable only by their type parameters (and the result type). Additionally, as shown below, the result type (`LambdaExpression`) of the second method signature is a subtype of that for the first one:

DRAFT

```
public static Expression<TDelegate> Lambda<TDelegate>(
       Expression body,
       params ParameterExpression[] parameters
)

public static LambdaExpression Lambda(
       Expression body,
       params ParameterExpression[] parameters
)
```

BTW, the above shows that .NET methods defining type args do not have their name mangled to account for the arity of those args, unlike names for generic types.

### 2.5 FYI: Visiting LINQ Expression Trees

`System.Linq.Expressions` Namespace (in `System.Core.dll`)

The API: `http://msdn.microsoft.com/en-us/library/system.linq.expressions.aspx` Quoting from `http://msdn.microsoft.com/en-us/library/bb397951.aspx`

> *In .NET Framework 4, the expression trees API also supports assignments and control flow expressions such as loops, conditional blocks, and try-catch blocks. By using the API, you can create expression trees that are more complex than those that can be created from lambda expressions by the C# and Visual Basic compilers. The following example demonstrates how to create an expression tree that calculates the factorial of a number.*

Expression trees are immutable and thus the need for a Cloning Visitor to obtain an updated version, as exemplified in `http://msdn.microsoft.com/en-us/library/bb546136.aspx`, based on a cloning subclass of `ExpressionVisitor` announced to be available at CodePlex `http://go.microsoft.com/fwlink/?LinkId=141028`.

TODO That URL is broken, find where that cloning visitor is located.

DRAFT

# 3 Planning ahead

### 3.1 Where to run the compiler plugin

The LINQ → SQO desugaring is not required to produce `Expression` nodes for “assignments and control flow expressions”. Also, cloning visitors are mentioned in Sec. 2.5 for completeness but actually no further modification of `Expression` trees is needed after being desugared from LINQ by (for example) a compiler plugin running after `parser` and before `namer`, as described next.

After adding new `Tree` subclasses to account for LINQ constructs, a modified parser (specific to Scala.NET) can produce trees containing “LINQ Tree nodes”, which have to be desugared (as per the recipe documented by Kaichuan Wen [1, Ch. 4]) before reaching the `namer` phase (otherwise, such nodes would constitute unexpected input to that phase). By having dedicated `Tree` classes, LINQ-level error messages and LINQ-level refactorings become possible (in the future).

### 3.2 The “old” vs. “new” syntax camps

The discussion in Sec. 3.1 suggests that LINQ support can be achieved as follows:

1. a modified parser returns instances of `Tree` subclasses (subclasses for LINQ-specific constructs)

2. normally no phase runs between `parser` and `namer`, but a custom phase (provided by a compiler plugin) can desugar those trees into SQO style

However, it could be argued that a compiler plugin is not needed, because the standard desguring of for-comprehensions can result in callsites to SQO wrappers, i.e. a form of “desugaring-by-library to SQO”, or “pimp my SQO”.

> Note: The `parser` phase desugars for-comprehensions into `foreach` invocations. After leaving `parser` there are no “for-comprehensions” anymore but a mixture of invocations on `map`, `filter`, and `flatMap`. Details about this in §1.2.3 (also in Listing 1.3) of
> `http://www.sts.tu-harburg.de/people/mi.garcia/ScalaCompilerCorner/UntanglingScalaASTs1ofN.pdf`

Even if the “new syntax” for LINQ brings no additional expressive power, it brings instantly many more users than all existing DB-query DSLs for Scala put together (the proposed “desugaring-by-library to SQO” would be just one more such DSL). So the compiler plugin is needed.

## 4 TODO: Lambdas

Scala.NET will translate `w => w.Length` into an instance of `scala.Function1`, while the LINQ framework expects instead an instance of ``System.Func`2``. For `w` a `String`, the lambda above has type `scala.Function1[String, Int]` on JVM-Scala.

I guess in the long run the above could be an instance of ``System.Func`2[String, Int32]`` on Scala.NET, but for now it’s not. So one would have to write the longish form one would have written in C# 2.0, before syntax for lambdas was introduced.

Some details about `scala.Function` classes:

```
/**
 *   Function with 1 parameter.
 *
 * In the following example the definition of
 *   succ is a shorthand for the anonymous class
 *   definition anonfun1:
 *
 *  object Main extends Application {
 *
 *    val succ = (x: Int) => x + 1
 *
 *    val anonfun1 = new Function1[Int, Int] {
 *      def apply(x: Int): Int = x + 1
 *    }
 *
 *    println(succ(0))
 *    println(anonfun1(0))
```

DRAFT

```
 *  }
 */
trait Function1[@specialized(scala.Int, scala.Long, scala.Float, scala.Double) -T1,
                @specialized(scala.Unit, scala.Boolean, scala.Int,
                             scala.Float, scala.Long, scala.Double) +R]
      extends AnyRef { self =>
  def apply(v1:T1): R
  override def toString() = "<function1>"

  /** (f compose g)(x) == f(g(x)) */
  def compose[A](g: A => T1): A => R = { x => apply(g(x)) }

  /** (f andThen g)(x) == g(f(x)) */
  def andThen[A](g: R => A): T1 => A = { x => g(apply(x)) }

}
```

# 5 Link Sink

- http://www.albahari.com/nutshell/linqkit.aspx
- C# Spec 4.0, http://www.microsoft.com/downloads/details.aspx?familyid=DFBF523C-F98C-4804-AFBD-459E846B268E&displaylang=en
- CSharpSourceReader. Quoting from http://cciast.codeplex.com/Thread/View.aspx?ThreadId=213903:

  > *I see in the tests there is a CSharpSourceEmitter and I can see a .felt file which appears to be a grammar for reading C# but I don't see any classes for actually doing the parsing? . . .*
  > *The grammar file was created for a parser generator that hasn't seen the light of day AFAIK. There is a partially complete parser for C# that produces CCI AST nodes, lurking in http://specsharp.codeplex.com under the SpecSharp2 directory.*

DRAFT

# References

[1] Kaichuan Wen. Translation of Java-embedded database queries, with a prototype implementation for LINQ, 2009. http://www.wen-k.com/files/JavaEmbeddedLinq.PDF.

Listing 1: Desugaring performed by the C# compiler

```
private static void Test4()
{
  ParameterExpression CS$0$0000;

  IQueryable<int> smallPrimes =
    new int[] { 1, 3, 13 }
   .AsQueryable<int>()
   .Where<int>
   (
      System.Linq.Expressions.Expression.Lambda<Func<int, bool>>
      (
             Expression.LessThan
             (
                 CS$0$0000 = Expression.Parameter(typeof(int), "q"),
                 Expression.Constant(11, typeof(int))
             ),
             new ParameterExpression[] { CS$0$0000 }
      )
    );

    Expression<Func<int, decimal?>> calcPrice =
       Expression.Lambda<Func<int, decimal?>>
       (
               Expression.Convert
               (
                        Expression.Multiply
                        (
                                Expression.Convert
                                (
                                        CS$0$0000 = Expression.Parameter(typeof(int), "p"),
                                        typeof(decimal),
                                        (MethodInfo) methodof(decimal.op_Implicit)
                                ),
                                Expression.Constant(1.19M, typeof(decimal))
                        ),
                        typeof(decimal?)
                ),
                new ParameterExpression[] { CS$0$0000 }
        );
}
```

DRAFT

Listing 2: A manually-performed expansion of LINQ as SQO queries

```
/*- IQueryable<int> */ smallPrimes =
  Array(1, 3, 13)  /*- i.e., Array[int], type argument inferred from elements */
 .AsQueryable/*- [int] */()
 .Where/*- [int] */
 (
     System.Linq.Expressions.Expression.Lambda<Func<int, bool>>
     /*- does the type arg above get inferred? Guess so but need to try out. */
     (
            Expression.LessThan
            (
                Expression.Parameter(typeof(int), "q"),
                Expression.Constant(11, Type.GetType("System.Int32")) /*- rather than typeof(int) */
            ),
            /*- can Type.GetType("System.Int32") below be provided implicitly?
                Say, if a subclass of Expression.Parameter is defined taking an implicit (and using it below)
                which invokes the base constructor passing that implicit.
             */
            Array/*- [ParameterExpression] */(Expression.Parameter(typeof(int), "q"))
     )
  );

  Expression<Func<int, decimal?>> calcPrice = /*- same question about inferring type arg as above */
    Expression.Lambda<Func<int, decimal?>> /*- same question here */
    (
        Expression.Convert
        (
          Expression.Multiply
          (
                   Expression.Convert
                   (
                            Expression.Parameter(Type.GetType("System.Int32"), "p"),
                            Type.GetType("System.Decimal"),
                            (MethodInfo) methodof(decimal.op_Implicit) /*- TODO */
                   ),
                   Expression.Constant(1.19M, typeof(decimal))
          ),
          typeof(decimal?)
         ),
         Array/*- [ParameterExpression] */(Expression.Parameter(typeof(int), "p"))
    );
```

DRAFT

DRAFT

Listing 3: See Sec. 2.4

```
type GList[T] = System.Collections.Generic.‘List\1401‘[T] /*- that’s how we save keyboard effort */
type Func2[T, TResult] = System.‘Func\1402‘[T, TResult]

def main(args: Array[String]) {

  import System.Linq.Expressions.Expression
  import System.Linq.Queryable

  val i32 = Type.GetType("System.Int32")
  val is = new GList[Int32](); // an Scala Array[Int] doesn’t yet implement System.Collections.IEnumerable
  val q1 = Queryable.AsQueryable(is)

  val aLessThanExpr = Expression.LessThan (
            Expression.Parameter(i32, "q"),
            Expression.Constant(11, i32)
        )

  /*- type inference would have worked */
  val arrOfExprParam : Array[System.Linq.Expressions.ParameterExpression] =
    Array(Expression.Parameter(i32, "q"))

  val lambdaExpr = System.Linq.Expressions.Expression.Lambda[Func2[Int32, Boolean]](
        aLessThanExpr,
        arrOfExprParam
   )
  val q2 = Queryable.Where(q1, lambdaExpr)
}
```

## Listing 4: After desugaring, CIL version, 1 of 2

```
.method private hidebysig static void Test4() cil managed
{
  // Code size       225 (0xe1)
  .maxstack 7
  .locals init ([0] int32 [] primes,
           [1] class [System.Core]System.Linq.IQueryable`1<int32> smallPrimes,
           [2] class [System.Core]System.Linq.Expressions.Expression`1<class [mscorlib]System.Func`2<int32,valuetype [mscorlib]System.Nullable`1<valuetype [mscorlib]System.Decimal>>> calcPrice,
           [3] class [System.Core]System.Linq.Expressions.ParameterExpression CS$0$0000,
           [4] class [System.Core]System.Linq.Expressions.ParameterExpression[] CS$0$0001)
  IL_0000:  nop
  IL_0001:  ldc.i4.3
  IL_0002:  newarr     [mscorlib]System.Int32
  IL_0007:  dup
  IL_0008:  ldtoken    field valuetype '<PrivateImplementationDetails>{7B54660E-1365-42BC-8B9F-26523307D31E}'/'__StaticArrayInitTypeSize=12' '<PrivateImplementationDetails>{7B54660E-1365-42BC-8B9F-26523307D31E}'::'$$method0x6000002-1'
  IL_000d:  call       void [mscorlib]System.Runtime.CompilerServices.RuntimeHelpers::InitializeArray(class [mscorlib]System.Array,
                                                                                                       valuetype [mscorlib]System.RuntimeFieldHandle)
  IL_0012:  stloc.0
  IL_0013:  ldloc.0
  IL_0014:  call       class [System.Core]System.Linq.IQueryable`1<!!0> [System.Core]System.Linq.Queryable::AsQueryable<int32>(class [mscorlib]System.Collections.Generic.IEnumerable`1<!!0>)
  IL_0019:  ldtoken    [mscorlib]System.Int32
  IL_001e:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_0023:  ldstr      "q"
  IL_0028:  call       class [System.Core]System.Linq.Expressions.ParameterExpression [System.Core]System.Linq.Expressions.Expression::Parameter(class [mscorlib]System.Type,
                                                                                                                                                    string)
  IL_002d:  stloc.3
  IL_002e:  ldloc.3
  IL_002f:  ldc.i4.s   11
  IL_0031:  box        [mscorlib]System.Int32
  IL_0036:  ldtoken    [mscorlib]System.Int32
  IL_003b:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_0040:  call       class [System.Core]System.Linq.Expressions.ConstantExpression [System.Core]System.Linq.Expressions.Expression::Constant(object,
                                                                                                                                                   class [mscorlib]System.Type)
  IL_0045:  call       class [System.Core]System.Linq.Expressions.BinaryExpression [System.Core]System.Linq.Expressions.Expression::LessThan(class [System.Core]System.Linq.Expressions.Expression,
                                                                                                                                                 class [System.Core]System.Linq.Expressions.Expression)
  IL_004a:  ldc.i4.1
  IL_004b:  newarr     [System.Core]System.Linq.Expressions.ParameterExpression
  IL_0050:  stloc.s    CS$0$0001
  IL_0052:  ldloc.s    CS$0$0001
  IL_0054:  ldc.i4.0
  IL_0055:  ldloc.3
  IL_0056:  stelem.ref
  IL_0057:  ldloc.s    CS$0$0001
  IL_0059:  call       class [System.Core]System.Linq.Expressions.Expression`1<!!0> [System.Core]System.Linq.Expressions.Expression::Lambda<class [mscorlib]System.Func`2<int32,bool>>(class [System.Core]System.Linq.Expressions.Expression,
                                                                                                                                                                                           class [System.Core]System.Linq.Expressions.ParameterExpression[])
  IL_005e:  call       class [System.Core]System.Linq.IQueryable`1<!!0> [System.Core]System.Linq.Queryable::Where<int32>(class [System.Core]System.Linq.IQueryable`1<!!0>,
                                                                                                                           class [System.Core]System.Linq.Expressions.Expression`1<class [mscorlib]System.Func`2<!!0,bool>>)
  IL_0063:  stloc.1
  IL_0064:  ldtoken    [mscorlib]System.Int32
  IL_0069:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_006e:  ldstr      "p"
  IL_0073:  call       class [System.Core]System.Linq.Expressions.ParameterExpression [System.Core]System.Linq.Expressions.Expression::Parameter(class [mscorlib]System.Type,
                                                                                                                                                    string)
  IL_0078:  stloc.3
  IL_0079:  ldloc.3
  IL_007a:  ldtoken    [mscorlib]System.Decimal
  IL_007f:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_0084:  ldtoken    method valuetype [mscorlib]System.Decimal [mscorlib]System.Decimal::op_Implicit(int32)
  IL_0089:  call       class [mscorlib]System.Reflection.MethodBase [mscorlib]System.Reflection.MethodBase::GetMethodFromHandle(valuetype [mscorlib]System.RuntimeMethodHandle)
  IL_008e:  castclass  [mscorlib]System.Reflection.MethodInfo
```

DRAFT

Listing 5: After desugaring, CIL version, 2 of 2

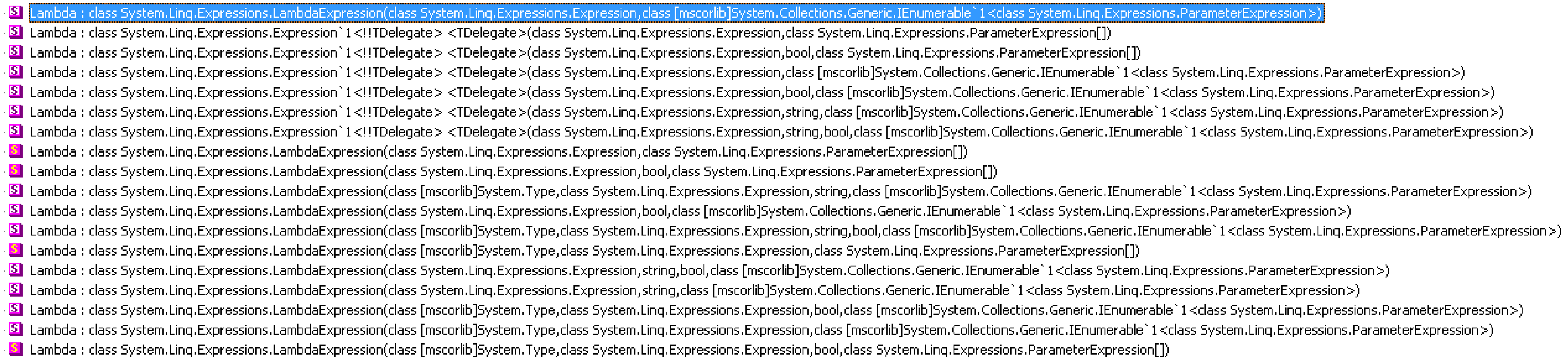
```
IL_0093:  call      class  [System.Core]System.Linq.Expressions . UnaryExpression  [System.Core]System.Linq.Expressions . Expression :: Convert( class  [System.Core]System.Linq.Expressions . Expression ,
                                                                                                                                               class  [ mscorlib ]System.Type,
                                                                                                                                               class  [ mscorlib ]System. Reflection . MethodInfo)
IL_0098:  ldc . i4 . s   119
IL_009a:  ldc . i4 .0
IL_009b:  ldc . i4 .0
IL_009c:  ldc . i4 .0
IL_009d:  ldc . i4 .2
IL_009e:  newobj    instance  void  [ mscorlib ]System.Decimal::. ctor ( int32 ,
                                                                  int32 ,
                                                                  int32 ,
                                                                  bool ,
                                                                  uint8 )
IL_00a3:  box       [ mscorlib ]System.Decimal
IL_00a8:  ldtoken   [ mscorlib ]System.Decimal
IL_00ad:  call      class  [ mscorlib ]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
IL_00b2:  call      class  [System.Core]System.Linq.Expressions . ConstantExpression  [System.Core]System.Linq.Expressions . Expression :: Constant(object ,
                                                                                                                                                  class  [ mscorlib ]System.Type)
IL_00b7:  call      class  [System.Core]System.Linq.Expressions . BinaryExpression  [System.Core]System.Linq.Expressions . Expression :: Multiply( class  [System.Core]System.Linq.Expressions . Expression ,
                                                                                                                                               class  [System.Core]System.Linq.Expressions . Expression )
IL_00bc:  ldtoken   valuetype  [ mscorlib ]System. Nullable'1 <valuetype [mscorlib]System.Decimal>
IL_00c1:  call      class  [ mscorlib ]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
IL_00c6:  call      class  [System.Core]System.Linq.Expressions . UnaryExpression  [System.Core]System.Linq.Expressions . Expression :: Convert( class  [System.Core]System.Linq.Expressions . Expression ,
                                                                                                                                               class  [ mscorlib ]System.Type)
IL_00cb:  ldc . i4 .1
IL_00cc:  newarr    [System.Core]System.Linq.Expressions . ParameterExpression
IL_00d1:  stloc . s   CS$0$0001
IL_00d3:  ldloc . s   CS$0$0001
IL_00d5:  ldc . i4 .0
IL_00d6:  ldloc .3
IL_00d7:  stelem . ref
IL_00d8:  ldloc . s   CS$0$0001
IL_00da:  call      class  [System.Core]System.Linq.Expressions . Expression'1 <!!0>
                    [System.Core]System.Linq.Expressions . Expression :: Lambda
                    <class [mscorlib ]System.Func'2
                      <int32,
                       valuetype  [ mscorlib ]System. Nullable'1
                       <valuetype [mscorlib]System.Decimal>
                      >
                    >
                      (
                        class  [System.Core]System.Linq.Expressions . Expression ,
                        class  [System.Core]System.Linq.Expressions . ParameterExpression []
                      )
IL_00df :  stloc .2
IL_00e0 :  ret
} // end of method Cba'2::Test4
```

DRAFT

## Listing 6: SQO Desugaring, CIL

```
.method private hidebysig static void Test5() cil managed
{
  // Code size       128 (0x80)
  .maxstack 5
  .locals init ([0] class [System.Core]System.Linq.IQueryable`1<int32> smallPrimes,
           [1] class [System.Core]System.Linq.Expressions.ParameterExpression CS$0$0000,
           [2] class [System.Core]System.Linq.Expressions.ParameterExpression[] CS$0$0001)
  IL_0000:  nop
  IL_0001:  ldsfld     int32[] class scala.tools.nsc.backend.msil.Cba`2<!T,!S>::primes
  IL_0006:  call       class [System.Core]System.Linq.IQueryable`1<!!0> [System.Core]System.Linq.Queryable::AsQueryable<int32>(class [mscorlib]System.Collections.Generic.IEnumerable`1<!!0>)
  IL_000b:  ldtoken    [mscorlib]System.Int32
  IL_0010:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_0015:  ldstr      "q"
  IL_001a:  call       class [System.Core]System.Linq.Expressions.ParameterExpression [System.Core]System.Linq.Expressions.Expression::Parameter(class [mscorlib]System.Type,
                                                                                                                                                   string)
  IL_001f:  stloc.1
  IL_0020:  ldloc.1
  IL_0021:  ldc.i4.s   11
  IL_0023:  box        [mscorlib]System.Int32
  IL_0028:  ldtoken    [mscorlib]System.Int32
  IL_002d:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_0032:  call       class [System.Core]System.Linq.Expressions.ConstantExpression [System.Core]System.Linq.Expressions.Expression::Constant(object,
                                                                                                                                                 class [mscorlib]System.Type)
  IL_0037:  call       class [System.Core]System.Linq.Expressions.BinaryExpression [System.Core]System.Linq.Expressions.Expression::LessThan(class [System.Core]System.Linq.Expressions.Expression,
                                                                                                                                               class [System.Core]System.Linq.Expressions.Expression)
  IL_003c:  ldc.i4.1
  IL_003d:  newarr     [System.Core]System.Linq.Expressions.ParameterExpression
  IL_0042:  stloc.2
  IL_0043:  ldloc.2
  IL_0044:  ldc.i4.0
  IL_0045:  ldloc.1
  IL_0046:  stelem.ref
  IL_0047:  ldloc.2
  IL_0048:  call       class [System.Core]System.Linq.Expressions.Expression`1<!!0> [System.Core]System.Linq.Expressions.Expression::Lambda<class [mscorlib]System.Func`2<int32,bool>>(class [System.Core]System.Linq.Expressions.Expression,
                                                                                                                                                                                          class [System.Core]System.Linq.Expressions.ParameterExpression[])
  IL_004d:  call       class [System.Core]System.Linq.IQueryable`1<!!0> [System.Core]System.Linq.Queryable::Where<int32>(class [System.Core]System.Linq.IQueryable`1<!!0>,
                                                                                                                       class [System.Core]System.Linq.Expressions.Expression`1<class [mscorlib]System.Func`2<!!0,bool>>)
  IL_0052:  ldtoken    [mscorlib]System.Int32
  IL_0057:  call       class [mscorlib]System.Type [mscorlib]System.Type::GetTypeFromHandle(valuetype [mscorlib]System.RuntimeTypeHandle)
  IL_005c:  ldstr      "q"
  IL_0061:  call       class [System.Core]System.Linq.Expressions.ParameterExpression [System.Core]System.Linq.Expressions.Expression::Parameter(class [mscorlib]System.Type,
                                                                                                                                                   string)
  IL_0066:  stloc.1
  IL_0067:  ldloc.1
  IL_0068:  ldc.i4.1
  IL_0069:  newarr     [System.Core]System.Linq.Expressions.ParameterExpression
  IL_006e:  stloc.2
  IL_006f:  ldloc.2
  IL_0070:  ldc.i4.0
  IL_0071:  ldloc.1
  IL_0072:  stelem.ref
  IL_0073:  ldloc.2
  IL_0074:  call       class [System.Core]System.Linq.Expressions.Expression`1<!!0> [System.Core]System.Linq.Expressions.Expression::Lambda<class [mscorlib]System.Func`2<int32,int32>>(class [System.Core]System.Linq.Expressions.Expression,
                                                                                                                                                                                           class [System.Core]System.Linq.Expressions.ParameterExpression[])
  IL_0079:  call       class [System.Core]System.Linq.IQueryable`1<!!1> [System.Core]System.Linq.Queryable::Select<int32,int32>(class [System.Core]System.Linq.IQueryable`1<!!0>,
                                                                                                                              class [System.Core]System.Linq.Expressions.Expression`1<class [mscorlib]System.Func`2<!!0,!!1>>)
  IL_007e:  stloc.0
  IL_007f:  ret
} // end of method Cba`2::Test5
```

DRAFT



```
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,class System.Linq.Expressions.ParameterExpression[])
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,bool,class System.Linq.Expressions.ParameterExpression[])
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,string,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.Expression`1<!!TDelegate> <TDelegate>(class System.Linq.Expressions.Expression,string,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,class System.Linq.Expressions.ParameterExpression[])
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,bool,class System.Linq.Expressions.ParameterExpression[])
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,string,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,string,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,class System.Linq.Expressions.ParameterExpression[])
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,string,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class System.Linq.Expressions.Expression,string,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,bool,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,class [mscorlib]System.Collections.Generic.IEnumerable`1<class System.Linq.Expressions.ParameterExpression>)
Lambda : class System.Linq.Expressions.LambdaExpression(class [mscorlib]System.Type,class System.Linq.Expressions.Expression,bool,class System.Linq.Expressions.ParameterExpression[])
```

Figure 1: LambdaOverloads

AFT