

SemanticDB

For Scala developer tools

About me

- Author of Scalafmt and Scalafix
- Software developer at Scala Center
- Scalameta maintainer



Credits

- Eugene Burmako:
Advanced Scala Tools
team lead at Twitter
- Co-maintainer of
Scalameta and
SemanticDB





A new not-for-profit center established at EPFL.

Focused on:

OPEN SOURCE

EDUCATION

Tooling

What is the Scala Center's Mission?

Scalameta

- Open Source Project with 50+ contributors
- Full-time developers funded by Scala Center + Twitter
- 120k downloads/month for core module



```
-Repositories.filter { t =>
  - (t.origin === repo.bind)
- }.sortBy(_.userName asc).list

+Repositories

+ .filter { t =>
+   (t.origin === repo.bind)
+
+ }
+ .sortBy(_.userName asc)
+ .list
```

scalafmt

```
- implicit val tt = liftedType  
+ implicit val tt: TypedType[R] = liftedType
```

scalafix

Rewrite and linting tool

We used Scalafix to perform a large-scale refactoring of our Scala codebase at Twitter with great success.

*– Shane Delmore,
Advanced Scala Tools Team @Twitter*

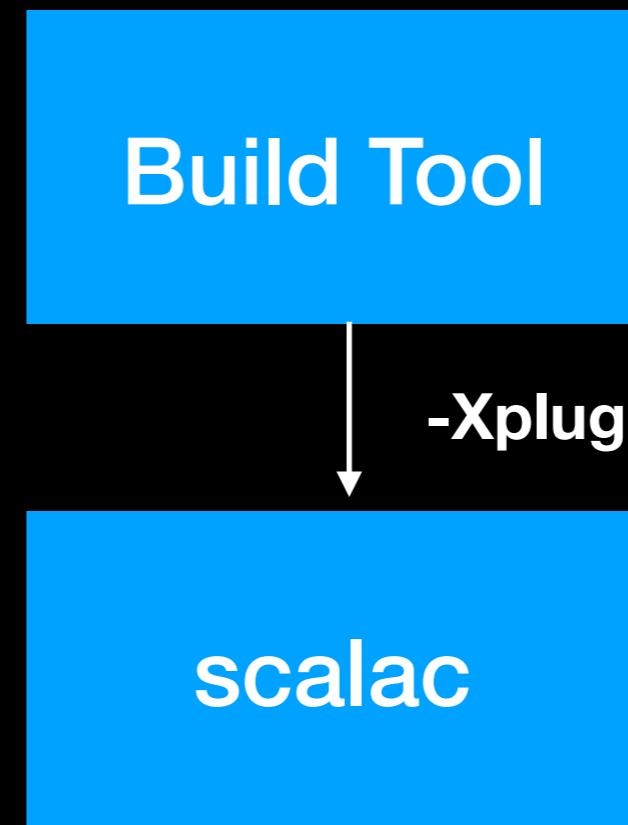
**Scalafix enabled us to run
refactorings in a distributed fashion,
which is necessary for a codebase
the size of ours.**

*- Shane Delmore,
Advanced Scala Tools Team @Twitter*

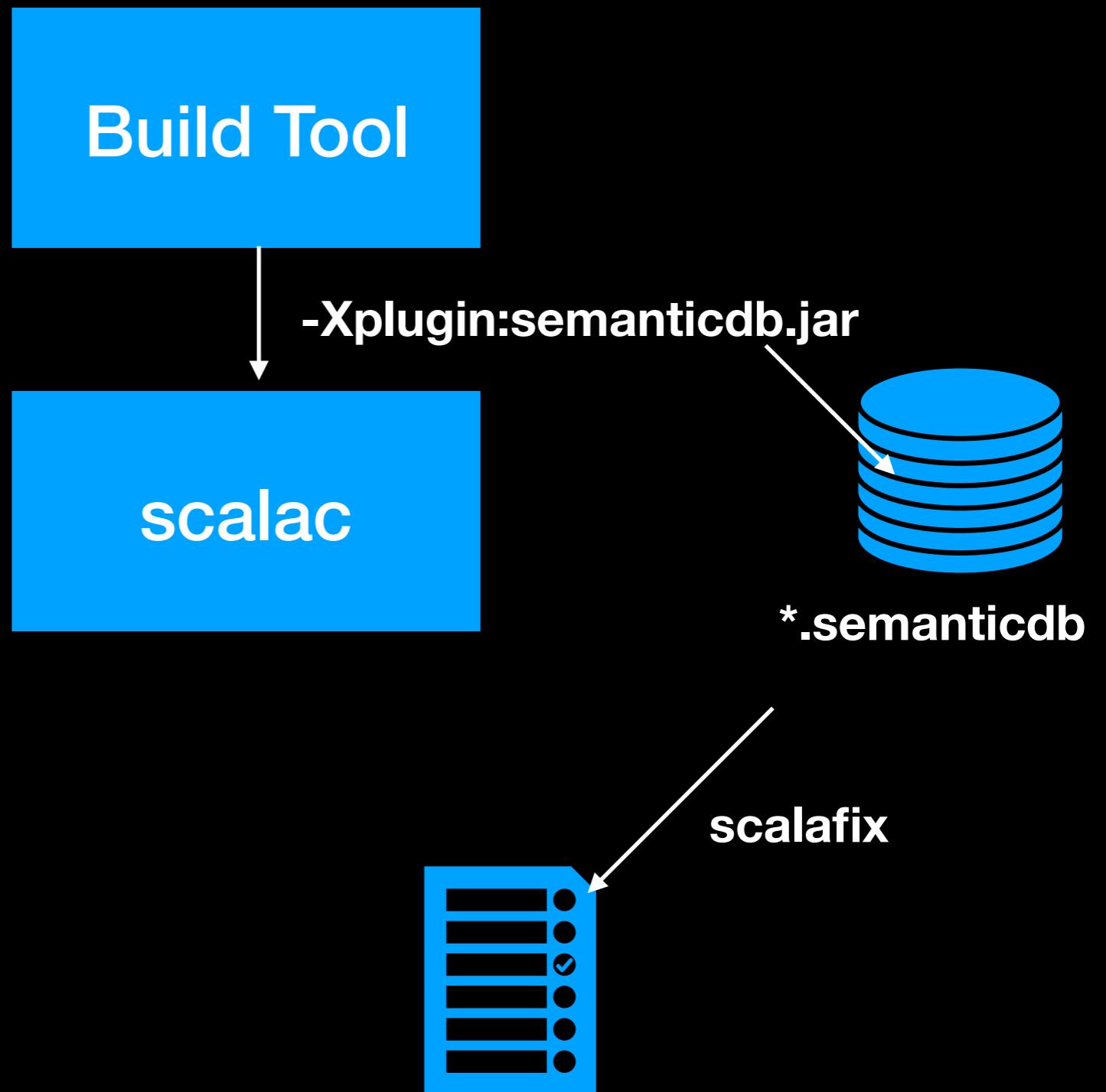
Agenda

- Compiler plugin vs. SemanticDB
- Schema
- Utilities
- Demo

Compiler plugin developer tools



SemanticDB developer tools



Compiler plugins

- Undocumented and unsupported internal APIs
- Relevant data available in different phases
- Analysis limited to single compilation run
- Difficult to test and maintain

SemanticDB

- 300 LOC protobuf schema: symbols, types, occurrences
- 11.000 word specification: Scala, Java
- Utilities: metacp, metac, mtags, metap
- Applications: metadoc, metals, scalafix

Schema

TextDocument

Uri: src/spire/math/Complex.scala

Schema: SEMANTICDB3

Text: “package spire.math\n\n@...

Language: SCALA

SymbolOccurrence: [

 scala.SerialVersionUID#
 spire.math.Complex#
 ...
]

SymbolInformation: [

 spire.math.Complex#
 spire.math.Complex#[T]
 ...
]

Diagnostics: [...]

Synthetics: [...]

Complex.scala ×

1 package spire.math
2
3 @SerialVersionUID(0L)

```
class Complex[@sp(Float, Double) T](real: T, imag: T)
  extends ScalaNumber
  with NumericConversions
  with Serializable { lhs =>

    def +(rhs: T)(implicit r: Semiring[T]): Complex[T] = 
      Complex(real + rhs, imag)

    def -(rhs: T)(implicit r: Rng[T]): Complex[T] = 
      Complex(real - rhs, imag)

    def *(rhs: T)(implicit r: Semiring[T]): Complex[T] = 
      Complex(real * rhs, imag * rhs)

    def /(rhs: T)(implicit r: Field[T]): Complex[T] = 
      Complex(real / rhs, imag / rhs)
```

SymbolOccurrence

Symbol: spire.math.Complex#
Range: 4:18-4:25
Role: DEFINITION

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T, imag: T)
5 extends ScalaNumber
6 with ScalaNumericConversions
7 with Serializable { lhs =>
8
9   def +(rhs: T)(implicit r: Semiring[T]): Complex[T] =
10    new Complex(real + rhs, imag)
11   def -(rhs: T)(implicit r: Rng[T]): Complex[T] =
12    new Complex(real - rhs, imag)
13   def *(rhs: T)(implicit r: Semiring[T]): Complex[T] =
14    new Complex(real * rhs, imag * rhs)
15   def /(rhs: T)(implicit r: Field[T]): Complex[T] =
16    new Complex(real / rhs, imag / rhs)
17
18 }
```

SymbolOccurrence

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T, imag: T)
```

Symbol: spire.math.Complex#real()

Range: 10:18-10:22

Role: REFERENCE

```
9     def +(rhs: T)(implicit r: Semiring[T]): Complex[T] =
10       new Complex(real + rhs, imag)
11     def -(rhs: T)(implicit r: Rng[T]): Complex[T] =
12       new Complex(real - rhs, imag)
13     def *(rhs: T)(implicit r: Semiring[T]): Complex[T] =
14       new Complex(real * rhs, imag * rhs)
15     def /(rhs: T)(implicit r: Field[T]): Complex[T] =
16       new Complex(real / rhs, imag / rhs)
17
18 }
```

SymbolInformation

Kind: CLASS
Language: SCALA
Name: Complex
Symbol: spire.math.Complex#
Owner: spire.math.
Type: ClassInfoType { +6 decls }
Properties: final, case
Annotations: scala.SerialVersionUID#
Accessibility: PUBLIC

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T, imag: T)
5   extends ScalaNumber
6   with ScalaNumericConversions
7   with Serializable { lhs =>
8
9     def +(rhs: T)(implicit r: Semiring[T]): Complex[T] =
10      Complex(real + rhs, imag)
11      def -(rhs: T)(implicit r: Rng[T]): Complex[T] =
12        Complex(real - rhs, imag)
13        def *(rhs: T)(implicit r: Semiring[T]): Complex[T] =
14          Complex(real * rhs, imag * rhs)
15        def /(rhs: T)(implicit r: Field[T]): Complex[T] =
16          new Complex(real / rhs, imag / rhs)
17
18 }
```

SymbolInformation

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T, imag: T)
5   extends ScalaNumber
6   with ScalaNumericConversions
7   with Serializable { lhs =>
8
9   def +(rhs: T)(implicit r: Semiring[T]): Complex[T] =
10    new Complex(real + rhs, imag)
11
12  def -(rhs: T)(implicit r: Rng[T]): Complex[T] =
13    new Complex(real - rhs, imag)
14
15  def *(rhs: T)(implicit r: Semiring[T]): Complex[T] =
16    new Complex(real * rhs, imag * rhs)
17
18  def /(rhs: T)(implicit r: Field[T]): Complex[T] =
19    new Complex(real / rhs, imag / rhs)
```

Kind: PARAMETER
Language: SCALA
Name: r
Symbol: spire.math.Complex#`+`(T,Semiring).(r)
Owner: spire.math.Complex#`+`(T,Semiring).
Type: TypeRef { Semiring[T] }
Properties: implicit

Spec

- Motivation
- Data Model
 - TextDocument
 - Language
 - URI
 - Range
 - Location
 - Symbol
 - Type
 - SymbolInformation
 - Annotation
 - Accessibility
 - SymbolOccurrence
 - Diagnostic
- Data Schemas
 - Protobuf
- Languages
 - Scala
 - Symbol
 - Type
 - SymbolInformation
 - Annotation
 - Accessibility
 - SymbolOccurrence
 - Java
 - Symbol
 - Type
 - SymbolInformation
 - Annotation

```

class C(p1: Int) {
  def m2(p2: Int) = ???
  def m3(p3: Int = 42) = ???
  def m4(p4: => Int) = ???
  def m5(p5: Int*) = ???
  def m6[T: C <% V] = ???
}

```

Definition	Symbol	Kind	Signature
p1	_empty_.C#`<init>`(Int).(p1)	PARAMETER	TypeRef(None, <Int>, List())
p2	_empty_.C#m2(Int).(p2)	PARAMETER	TypeRef(None, <Int>, List())
p3	_empty_.C#m3(Int).(p3)	PARAMETER	TypeRef(None, <Int>, List())
m3\$default\$1	_empty_.C#m3\$default\$1()	METHOD	TypeRef(None, <Int>, List())
p4	_empty_.C#m4(=>Int).(p4)	PARAMETER	ByNameType(TypeRef(None, <Int>, List()))
p5	_empty_.C#m5(Int*).(p5)	PARAMETER	RepeatedType(TypeRef(None, <Int>, List()))
Context bound	_empty_.C#m6(C,V).(x\$1)	PARAMETER	TypeRef(None, <C>, List(<T>))
View bound	_empty_.C#m7(C,V).(x\$2)	PARAMETER	TypeRef(None, <Function1>, List(<T>, <V>))

Scala parameters

```

class C extends S1 implements I {
    T1 m1;
    static T2 m2();
    T3 m3(one.Overload e1);
    static T4 m3(two.Overload e2);
    T5 m3(three.Overload e3);
    static class D1<T6 extends S2 & S3, T7> { }
    class D2 { }
}

```

Definition	Symbol	Kind	Signature
C	a.C#	CLASS	ClassInfoType(List(), List(<S1>, <I>), List(<m1>, <m2>, <m3(Overload)>, <m3(Overload+1)>, <m3(Overload+2)>, <D1>, <D2>))
m1	a.C#m1.	FIELD	TypeRef(None, <T1>, List())
m2	a.C#m2().	METHOD	MethodType(List(), List(), TypeRef(None, <T2>, List()))
m3	a.C#m3(Overload).	METHOD	MethodType(List(), List(<e1>), TypeRef(None, <T3>))
e1	a.C#m3(Overload).(e1)	PARAMETER	TypeRef(None, <one.Overload>, List())
m3	a.C#m3(Overload+1).	METHOD	MethodType(List(), List(<e3>), TypeRef(None, <T5>))

Java class

\$ metac

\$ metacp

\$ mtags

\$ metap

Utilities

metac

Build SemanticDB with compiler

```
$ scalac -Xplugin:semanticdb.jar  
         -Xstop-after:semanticdb-typer  
src/Complex.scala
```



```
$ metac src/Complex.scala
```



TextDocument

```
Complex.scala x
1 package spire.math
2
3 @SerialVersionUID(0L)
Uri: src/spire/math/Complex.scala      class Complex[@sp(Float, Double) T](real: T,
Schema: SEMANTICDB3                      ScalaNumber
Text: "package spire.math\n\n@..."          ScalaNumericConversions
Language: SCALA                           Serializable { lhs =>
SymbolOccurrence: [                        : T)(implicit r: Semiring[T]): Complex[T] =
  scala.SerialVersionUID#                  Complex(real + rhs, imag)
  spire.math.Complex#                     Complex(real - rhs, imag)
  ...                                     Complex(real * rhs, imag * rhs)
]                                         Complex(real / rhs, imag / rhs)
SymbolInformation: [                        ...
  spire.math.Complex#                    ...
  spire.math.Complex#[T]                 ...
]                                         ...
Diagnostics: [...]
Synthetics: [...]
```

src
└── Complex.scala

└── Complex.scala.semanticdb

metacp

Build SemanticDB with classpath

```
def process(  
    classpath: Classpath, settings: Setting  
) : Option[Classpath]
```

```
$ metacp scala-library.jar  
/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-356A4B.jar
```

```
def process(  
    classpath: Classpath, settings: Setting  
) : Option[Classpath]
```

```
$ metacp scala-library.jar  
/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-356A4B.jar
```

SymbolInformation

Kind: CLASS
Language: SCALA
Name: Complex
Symbol: spire.math.Complex#
Owner: spire.math
Type: ClassInfoType { +6 decls }
Properties: final, case
Annotations: scala.SerialVersionUID#
Accessibility: PUBLIC

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T,
5   | extends ScalaNumber
6   |   scalaNumericConversions
7   |   Serializable { lhs =>
8     |     rhs: T)(implicit r: Semiring[T]): Complex[T] =
9     |       Complex(real + rhs, imag)
10    |     rhs: T)(implicit r: Rng[T]): Complex[T] =
11      |       Complex(real - rhs, imag)
12    |     rhs: T)(implicit r: Semiring[T]): Complex[T] =
13      |       Complex(real * rhs, imag * rhs)
14
15    def /(rhs: T)(implicit r: Field[T]): Complex[T] =
16      new Complex(real / rhs, imag / rhs)
17
18 }
```

\$ metacp scala-library.jar

/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-356A4B.jar

mtags

Build SemanticDB with parser

```
def process(  
    classpath: Classpath): Option[Classpath]
```

```
$ mtags scala-library-sources.jar  
/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-sources-356A4B.jar
```

```
def process(  
    classpath: Classpath): Option[Classpath]
```

```
$ mtags scala-library-sources.jar  
/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-sources-356A4B.jar
```

SymbolOccurrence

Symbol: spire.math.Complex#
Range: 4:18-4:25
Role: DEFINITION

```
1 package spire.math
2
3 @SerialVersionUID(0L)
4 final case class Complex[@sp(Float, Double) T](real: T, imag: T)
5 extends ScalaNumber
6 with ScalaNumericConversions
7 with Serializable { lhs =>
8
9   def +(rhs: T)(implicit r: Semiring[T]): Complex[T] =
10    new Complex(real + rhs, imag)
11   def -(rhs: T)(implicit r: Rng[T]): Complex[T] =
12    new Complex(real - rhs, imag)
13   def *(rhs: T)(implicit r: Semiring[T]): Complex[T] =
14    new Complex(real * rhs, imag * rhs)
15   def /(rhs: T)(implicit r: Field[T]): Complex[T] =
16    new Complex(real / rhs, imag / rhs)
17
18 }
```

\$ mtags scala-library-sources.jar

/Users/ollie/Library/Caches/semanticdb/3.7.2/scala-library-sources-356A4B.jar

metap

Pretty-print SemanticDB

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27): <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27): <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}      metacp
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27):  <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27):  <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27):  <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

mtags

```
$ metap META-INF/semanticdb/Complex.scala.semanticdb
```

Symbols:

```
spire.math.Complex# => final case class Complex[T >: Nothing <: Any].{+16 decls}
  extends AnyRef
  extends Product
  extends Serializable
spire.math.Complex#`<init>`(T,T). => primary ctor <init>: (val real: T, val imag: T): Complex[T]
  real => spire.math.Complex#`<init>`(T,T).(real)
  T => spire.math.Complex#[T]
  imag => spire.math.Complex#`<init>`(T,T).(imag)
  Complex => spire.math.Complex#
```

Occurrences:

```
[0:8..0:13): spire => spire.
[0:14..0:18): math => spire.math.
[1:17..1:24): Complex <= spire.math.Complex#
[1:25..1:26): T <= spire.math.Complex#[T]
[1:27..1:27):  <= spire.math.Complex#`<init>`(T,T).
[1:28..1:32): real <= spire.math.Complex#real().
[1:34..1:35): T => spire.math.Complex#[T]
[1:37..1:41): imag <= spire.math.Complex#imag().
[1:43..1:44): T => spire.math.Complex#[T]
```

metac

Demo

metadoc, metals, scalafix

metadoc

Static site code browser with IDE-features

metals

Scala language server

scalafix

Refactoring and linting tool

Conclusion

- 300 LOC protobuf schema: symbols, types, occurrences
- 11.000 word specification: Scala, Java
- Utilities: metac, metacp, mtags, metap
- Applications: metadoc, metals, scalafix