scalafmt: opinionated code formatter for Scala

Ólafur Páll Geirsson

École Polytechnique Fédérale de Lausanne
School of Computer and Communication Sciences

July 7, 2016

Today's agenda

Overview

What is code formatting?

Unformatted

```
object    MyApp
  extends App {
  Initialize  ( context, config(port(
    "port.http"),
   settings + custom))
  Loop(  )
}
```

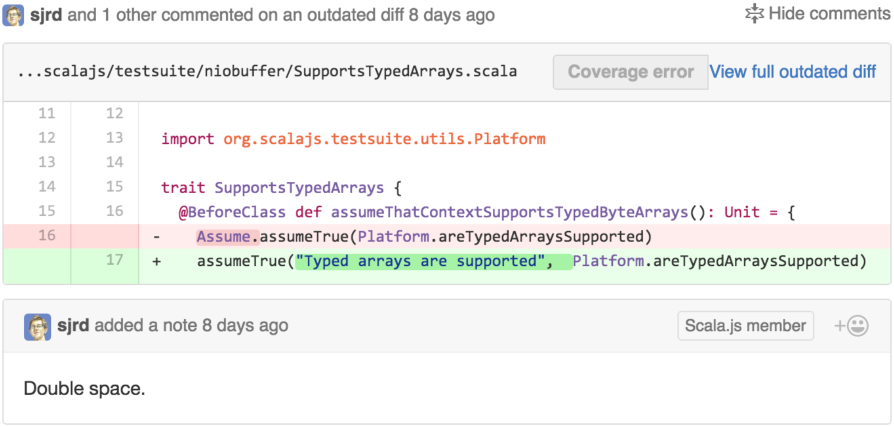
What is code formatting?

Formatted

```
object MyApp extends App {
  Initialize(
      context,
      config(port("port.http"),
             settings + custom))
  Loop()
}
```

Why?

Reason 1: Collaborative environments


sjrd and 1 other commented on an outdated diff 8 days ago

Hide comments

...scalajs/testsuite/niobuffer/SupportsTypedArrays.scala

Coverage error View full outdated diff

```
11  12
12  13   import org.scalajs.testsuite.utils.Platform
13  14
14  15   trait SupportsTypedArrays {
15  16     @BeforeClass def assumeThatContextSupportsTypedByteArrays(): Unit = {
16     -     Assume.assumeTrue(Platform.areTypedArraysSupported)
    17 +     assumeTrue("Typed arrays are supported", Platform.areTypedArraysSupported)
```

sjrd added a note 8 days ago

Scala.js member

Double space.

Reason 2: Refactoring

Large-Scale Automated Refactoring Using ClangMR[1]

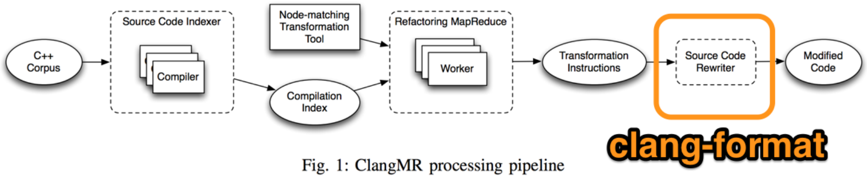


Fig. 1: ClangMR processing pipeline

[1]Source: http://research.google.com/pubs/pub41342.html

Problem statement

What *algorithms* and *data structures* allow us to develop a Scala code formatter with the following features?

- *Maximum line length setting*
- *Opinionated setting*
- *Vertical alignment*
- *Good performance*

Maximum line length setting

```scala
// 40 character max line length         |
object MyApp extends App {
  // BAD
  Initialize(context, config(port("port.http"),
    settings + custom))

  // OK
  Initialize(
      context,
      config(port("port.http"),
             settings + custom))
}
```

Opinionated setting

My definition

Disregard line breaking decisions in the original source to ensure that formatted sources follow a uniform coding style.

```
// Bin-pack
class Point(val x: Int, val y: Int,
    val z: Int)

// No bin-pack
class Point(val x: Int,
            val y: Int,
            val z: Int)
```

Vertical alignment

```
object VerticalAlignment {
  x match {
    case 1  => 1  -> 2  // first
    case 11 => 11 -> 22 // second
  }

  def name   = column[String]("name")
  def status = column[Int]("status")

  libraryDependencies ++= Seq(
    "org.scala-lang" % "scala-compiler" % "2.11.7",
    "com.lihaoyi"   %% "sourcecode"     % "0.1.1"
  )
}
```

Performance

- IDEs: reformat file on save
- Build tools: reformat file on compile
- Continuous integration: reformat diff before code review

Overview

Scalariform

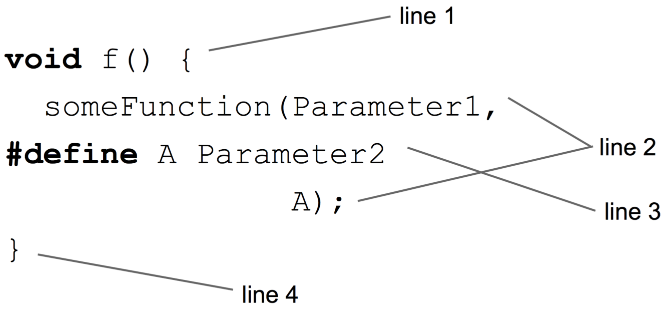
- No maximum line length setting
- No opinionated setting

ClangFormat[2]

[2]Source: `https://www.youtube.com/watch?v=s7JmdCfI__c`

Parser

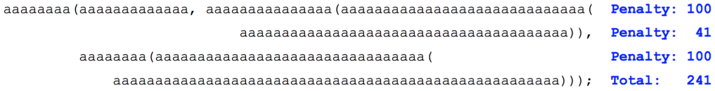
- Custom *UnwrappedLine* parser for C, C++, Objective-C, Java, JavaScript and Protobuf
  - handles invalid code code
  - ∼4.000 LOC



```
void f() {
  someFunction(Parameter1,
#define A Parameter2
                 A);
}
```

Line breaking: shortest path search

- Dijkstra’s shortest path for optimal line breaking.
  - Non-whitespace tokens are nodes
  - Whitespace tokens are edges



```
aaaaaaaa(aaaaaaaaaaaaa, aaaaaaaaaaaaaaa(aaaaaaaaaaaaaaaaaaaaaaaaaaaaa(  Penalty: 100
                            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)),  Penalty:  41
         aaaaaaaa(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa(                     Penalty: 100
             aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)));  Total:   241
```

rfmt[3]

[3]Source: `http://research.google.com/pubs/pub44667.html`

Formatting algebra

- Three layout operators

```
Lorem ipsum dolor
```

$'txt'$

```
Lorem ipsum dolor
consectetur adipiscing elit
```

$l_1 \updownarrow l_2$

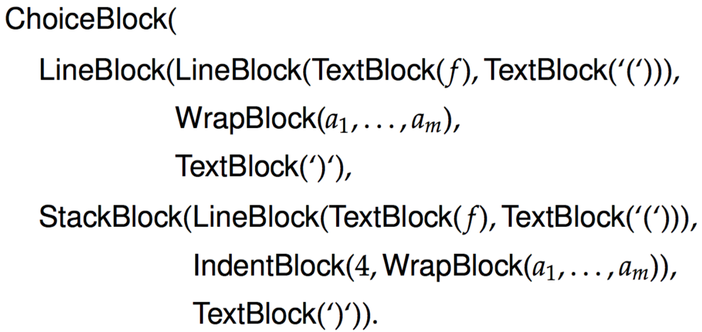
```
Lorem ipsum dolor
consectetur adipiscing elit Aliquam erat volutpat
                            condimentum vitae leo sit
```

$l_1 \leftrightarrow l_2$

- one *choice* operator “?”

Translating R to formatting algebra

- Custom R parser
  - ∼1.000 LOC
  - Comments are AST nodes
- “Block language” implemented in terms of primitive combinators



ChoiceBlock(

  LineBlock(LineBlock(TextBlock(f), TextBlock(‘(’)),

   WrapBlock($a_1, \ldots, a_m$),

   TextBlock(‘)’),

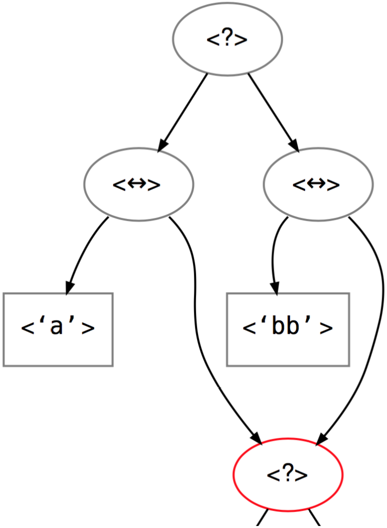
  StackBlock(LineBlock(TextBlock(f), TextBlock(‘(’)),

   IndentBlock(4, WrapBlock($a_1, \ldots, a_m$)),

   TextBlock(‘)’)).

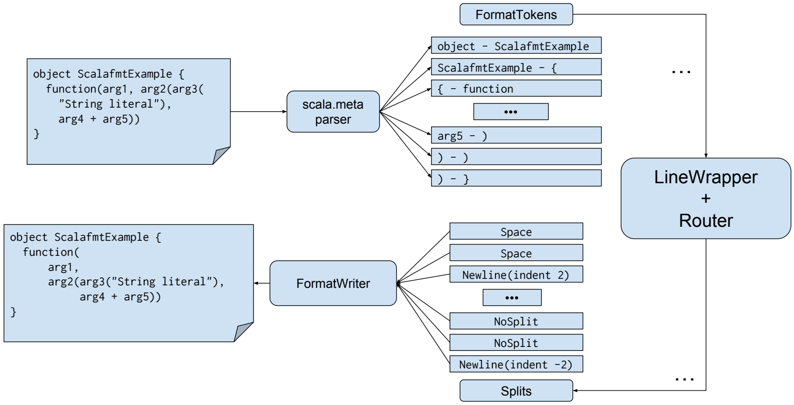
Line breaking: dynamic programming

- Dynamic programming to find optimal line breaking
  - (AST node, column) pairs are keys
  - can extrapolate missing columns



Overview

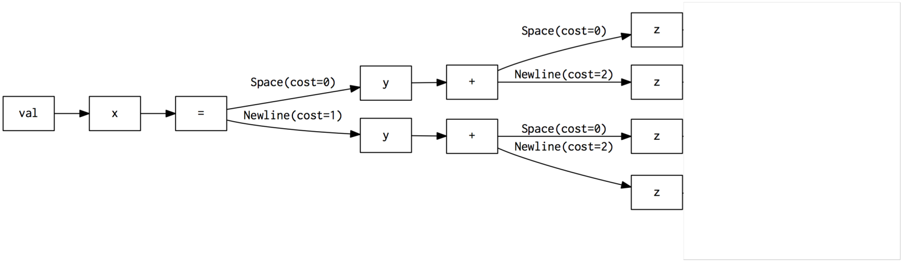
Architecture



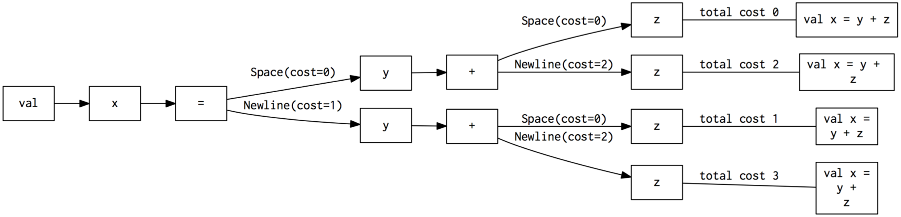
```
object ScalafmtExample {
  function(arg1, arg2(arg3(
    "String literal"),
    arg4 + arg5))
}
```

```
object ScalafmtExample {
  function(
      arg1,
      arg2(arg3("String literal"),
          arg4 + arg5))
}
```

LineWrapper + Router



LineWrapper + Router



Router

- One big pattern match ($\sim$ 1.100 LOC) on pairs of tokens

```
case FormatToken(_: Keyword, _) => Seq(Split(Space, 0))
case FormatToken(_, _: `=`)      => Seq(Split(Space, 0))
case FormatToken(_: `=`, _)      => Seq(Split(Space, 0)
                                        Split(Newline, 1))
// ...
```

Naïve best-first search

- Small: $\sim$ 20 LOC
- Exponential running time for basic programs

Optimization 1: dequeueOnNewStatements

```
def x = {
  function1(argument1, argument2, argument3)
  function2(argument1, argument2, argument3)
}
```

Optimization 2: OptimalToken

```
Database(UserObject(name1, age1),
         UserObject(name2, age2),
         // ...
         UserObject(nameN, ageN))
```

Optimization 3: escapeInPathologicalCases

- Give up, default behavior
- Best-effort, `-bestEffortInDeeplyNestedCode`

```
Defn.Object(Nil, Term.Name("State"), Template(Nil,
   Seq(Ctor.Ref.Name("Logger")), Term.Param(Nil,
   Name.Anonymous(), None, None),
   Some(Seq(Defn.Val(Nil,
   Seq(Pat.Var.Term(Term.Name("start"))), None,
   Term.Apply(Term.Name("State"), Seq())),
   Defn.Def( /* ... */)))))
```

Vertical alignment

- Implementation: 130 LOC imperative
- Running time: linear
- Configuration: `-alignTokens =>;Case,->;Infix,//;.*`

```
x match {
  case 1  => 1  -> 2  // first
  case 11 => 11 -> 22 // second
}
```

Summary: algorithms

Component	Lines of code
Router	1.070
FormatWriter	175
Best-first search	369
Utilities and data structures	1.547
Total	3.161

Testing?

Property 1: can format

```
forAll { code =>
  whenever(scalaCompilerCanParse(code)) {
    format(code).isInstanceOf[Success]
  }
}
```

Property 2: can format

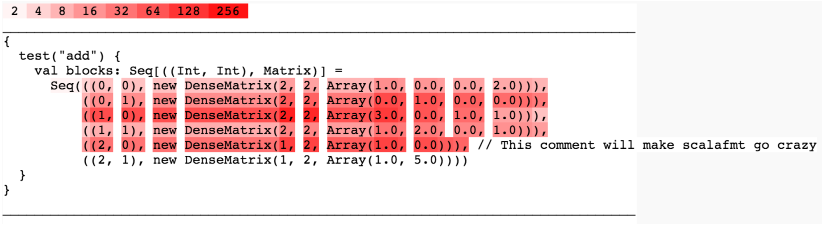
```
forAll { code =>
  ast(code) == ast(format(code))
}
```

Property 3: idempotent

```
forAll { code =>
  format(code) == format(format(code))
}
```

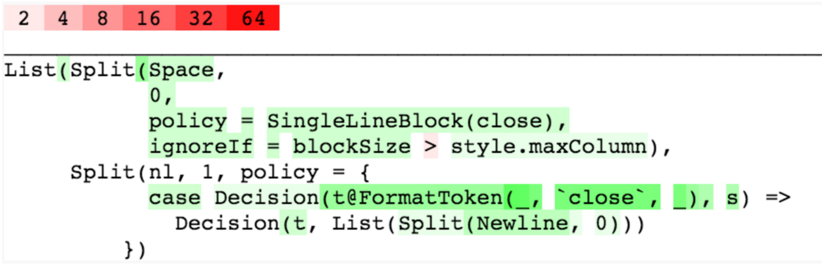
Tooling

Heatmap



```
2  4  8  16  32  64  128  256
_______________________________________________________________________________
{
  test("add") {
    val blocks: Seq[((Int, Int), Matrix)] =
      Seq(((0, 0), new DenseMatrix(2, 2, Array(1.0, 0.0, 0.0, 2.0))),
          ((0, 1), new DenseMatrix(2, 2, Array(0.0, 1.0, 0.0, 0.0))),
          ((1, 0), new DenseMatrix(2, 2, Array(3.0, 0.0, 1.0, 1.0))),
          ((1, 1), new DenseMatrix(2, 2, Array(1.0, 2.0, 0.0, 1.0))),
          ((2, 0), new DenseMatrix(1, 2, Array(1.0, 0.0))), // This comment will make scalafmt go crazy
          ((2, 1), new DenseMatrix(1, 2, Array(1.0, 5.0))))
  }
}
_______________________________________________________________________________
```

Diff heatmap



```
2  4  8  16  32  64

List(Split(Space,
           0,
           policy = SingleLineBlock(close),
           ignoreIf = blockSize > style.maxColumn),
     Split(nl, 1, policy = {
           case Decision(t@FormatToken(_, `close`, _), s) =>
             Decision(t, List(Split(Newline, 0)))
         })
```

Overview

Macro benchmark

Insight

How does scalafmt perform in a continuous integration setup?

Task

Format Scala.js repo.

Macro benchmark

Benchmark	Cores	Score	Error	Units
Parallel.scalafmt	4	14.616	0.632	s/op
Parallel.scalariform	4	2.810	0.641	s/op
Ratio		5.20		
Synchronous.scalafmt	1	35.654	0.459	s/op
Synchronous.scalariform	1	5.951	0.135	s/op
Ratio		5.99		

Micro benchmark

Insight

How does scalafmt perform in an interactive software developer workflow?

Task

Format single source file.

File sizes

Table: Lines of code per source file. Collected from sample of ∼27.000 source files with total 3.2 million lines of code.

25th	Median	Mean	75th	90th	95th	99th	Max
16	46	106	113	248	400	945	11.723

- Small: ∼ 50 LOC
- Medium: ∼ 300 LOC
- Large: ∼ 1.000 LOC
- Extra large: ∼ 4.500 LOC

Micro benchmark: results

Benchmark	Score	Error	Units
Small.scalafmt	6.968	0.104	ms/op
Small.scalariform	1.176	0.025	ms/op
Ratio	5.93		
Medium.scalafmt	79.616	2.013	ms/op
Medium.scalariform	15.934	0.441	ms/op
Ratio	5.00		

Micro benchmark: results

Benchmark	Score	Error	Units
Large.scalafmt	355.819	17.385	ms/op
Large.scalariform	39.324	3.395	ms/op
Ratio	9.05		
ExtraLarge.scalafmt	1423.140	103.360	ms/op
ExtraLarge.scalariform	219.820	14.450	ms/op
Ratio	6.50		

Installations

Table: Download numbers for scalafmt

Channel	Version	Installations
IntelliJ	v0.2.5	847
	All	3.273
Maven	v0.2.5	788
	All	2.657
Github	v0.2.5	102
	All	929
Sum	v0.2.5	1.737
	All	6.859

Installations

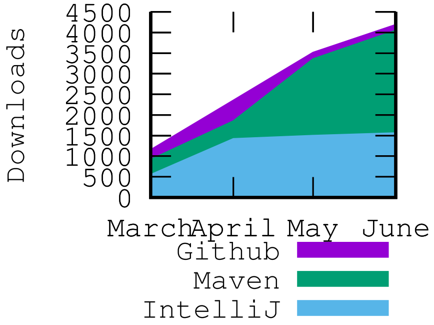


Figure: Scalafmt installations by month by channel

Overview

Conclusions

Maximum line length setting	X
Opinionated settings	X
Vertical alignment	X
Performance	?[4]

[4] Seems many users are OK with current performance

Verizon

"Verizon is now including scalafmt (with reformat on compile settings) in the default template for all new projects (which, in a sizable microservices shop, is a lot of projects)"

— Daniel Spiewak[5]

[5]Source:
`https://gitter.im/olafurpg/scalafmt?at=5776b518cdab7a1f4fbefd31`

The End

Thank you!