scalafmt: opinionated code formatter for Scala

Ólafur Páll Geirsson

École Polytechnique Fédérale de Lausanne
School of Computer and Communication Sciences

July 5, 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sjrd and 1 other commented on an outdated diff 8 days ago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View full outdated diff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]

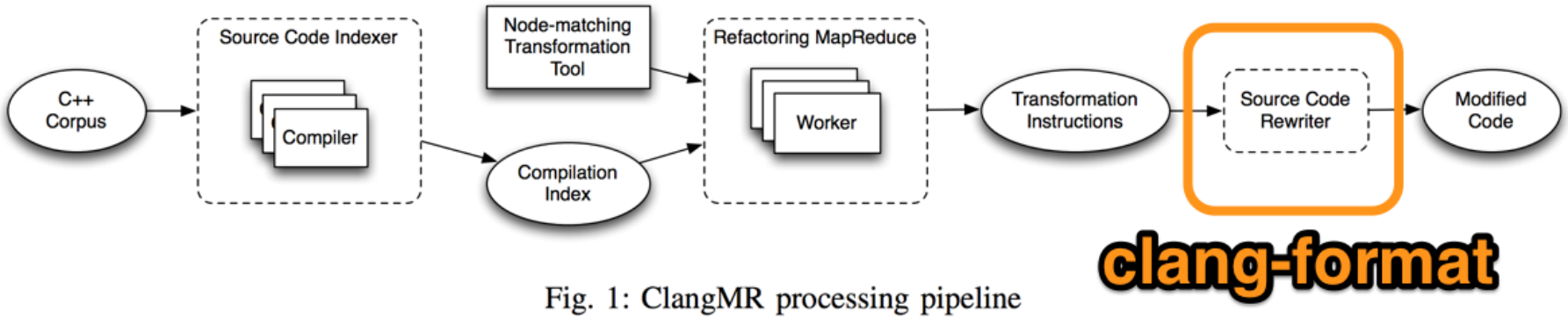


Fig. 1: ClangMR processing pipeline

[1] Wright et al. 2013.

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

```
// 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

Parser

- Custom *UnwrappedLine* parser for C, C++, Objective-C, Java, JavaScript and Protobuf[2]
  - handles invalid code code
  - ∼4.000 LOC

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

line 1
line 2
line 3
line 4

[2] **jasper_clang-format_2014**.

Line breaking: shortest path search

- Dijkstra's shortest path for optimal line breaking.[3]
  - Non-whitespace tokens are nodes
  - Whitespace tokens are edges

```
aaaaaaaa(aaaaaaaaaaaaa, aaaaaaaaaaaaaaa(aaaaaaaaaaaaaaaaaaaaaaaaaaaaa(   Penalty: 100
                                aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)),  Penalty:  41
            aaaaaaaa(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa(                     Penalty: 100
                aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa));  Total:   241
```

[3] **jasper_clang-format_2014**.

rfmt

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 "?"

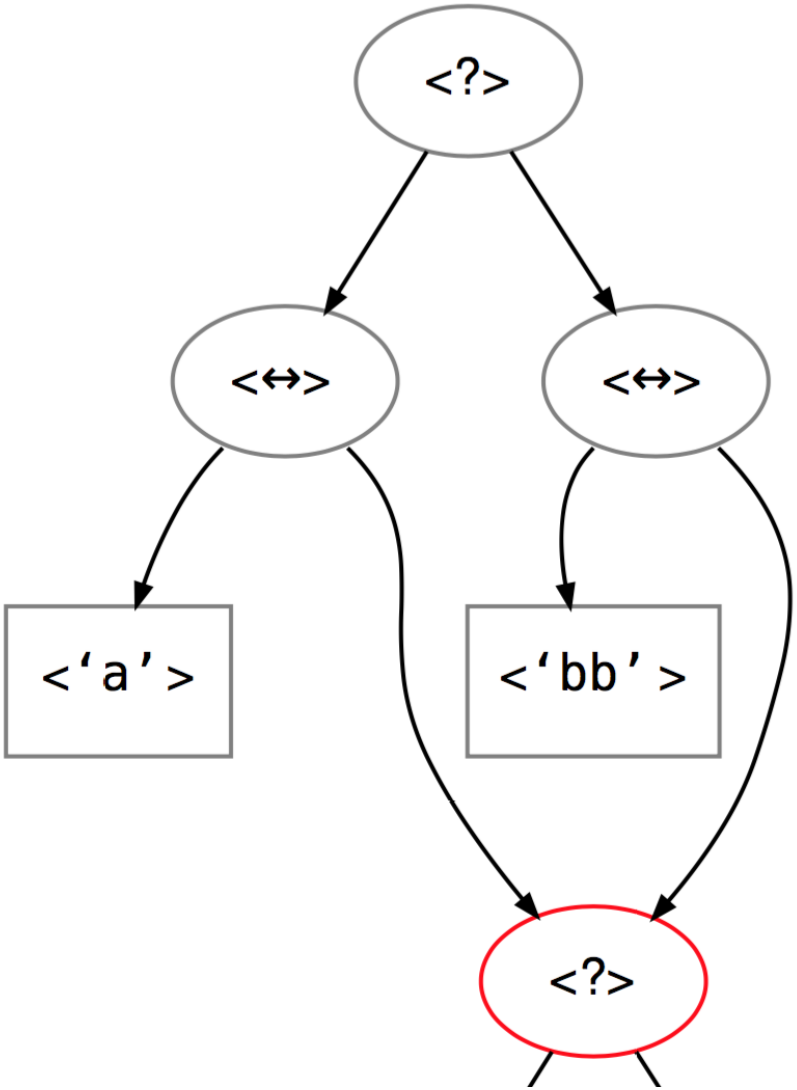
Translating R to formatting algebra

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

$$
\begin{aligned}
&\text{ChoiceBlock}(\\
&\quad \text{LineBlock}(\text{LineBlock}(\text{TextBlock}(f), \text{TextBlock}(`(`))),\\
&\qquad\qquad \text{WrapBlock}(a_1, \ldots, a_m),\\
&\qquad\qquad \text{TextBlock}(`)`),\\
&\quad \text{StackBlock}(\text{LineBlock}(\text{TextBlock}(f), \text{TextBlock}(`(`))),\\
&\qquad\qquad \text{IndentBlock}(4, \text{WrapBlock}(a_1, \ldots, a_m)),\\
&\qquad\qquad \text{TextBlock}(`)`)).
\end{aligned}
$$

Line breaking: dynamic programming

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



Overview

Title

- bullet

Example split

```
case FormatToken(_, els: `else`, _) =>
  val expire = owners(els).tokens.last
  Seq(
      Split(Space, 0, ignoreIf = newlines > 0)
        .withOptimalToken(expire)
        .withPolicy(SingleLineBlock(expire)),
      Split(Newline, 1)
  )
```

Overview

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

Overview

Conclusion

Scalafmt

- ???

The End

Thank you!

References

Wright, Hyrum et al. (2013). “Large-Scale Automated Refactoring Using ClangMR”. In: URL: https://research.google.com/pubs/pub41342.html (visited on 04/21/2016).