xfmt: yet another approach to code formatting and its applications to Scala.

Ólafur Páll Geirsson

School of Computer and Communication Sciences

Master's Thesis

June 2015

Responsible
Prof. Martin Odersky
EPFL / LAMP

Supervisor
Eugene Burmako
EPFL / LAMP

Abstract

Automatic code formatters bring many benefits to software development. When done right, code formatters relieve the developer's attention from manipulating syntactic trivia while enforcing a consistent coding style between teams. Still, little research has been made towards the algorithms and tools that make it possible to develop such code formatters.

This thesis addresses the problem of developing an advanced code formatter for a custom programming language. Our contributions are twofold. First, we present xmft, a language agnostic framework that consists of core data structures, algorithms and tooling that allow rapid development and testing of such a code formatter. Secondly, we provide a implementation of a code formatter, `scalafmt`, that uses our framework.

We show that the framework can support rich formatting options such as column-width limitation and configurable vertical alignment. In one month, the formatter has been installed over 2.000 times and XXX success metric.

Contents

1 Introduction

1.1 Problem statement

1.2 Contributions

The main contributions presented in this thesis are the following:

- Data structures, algorithms and tools for implementing advandes code formatters for syntatically rich languages.
- scalafmt, a case study where the framework is used to format Scala programs.

Non-goals:

- Language independent pretty printing. Hasn't gained popularity.

2 Background

1. Optimal line breaking
2. Oppen
3. rustfmt

2.1 Combinator based

1. Houghes 1995
2. Wadler 1999

2.2 Optimization-oriented

1. clang-format Dijkstra's 2010
2. dartfmt Best-first search 2014
3. rfmt 2015

1. AST integrity
2. Idempotent

3 Framework

3.1 Data structures

1. FormatToken.
2. Split.
3. State.
4. Indent.

3.2 Algorithms

1. Policy.
2. OptimalToken.
3. Best first search.

3.3 Tooling

4 scalafmt

4.1 scala.meta

4.2 Extensions

4.2.1 Optimizations

1. dequeueOnNewStatements
2. recurseOnBlocks
3. escapeInPathologicalCases
4. pruneSlowStates

5 Discussion

5.1 Future work

5.2 Conclusion