Imperial College
London

MEng Individual Project:
Interim Report

Imperial College London

Department of Computing

Automated Parser Combinator Linting and Refactoring Tools

Supervisor:
Dr. Jamie Willis

Author:
Rocco Jiang

Second Marker:
Dr. Robert Chatley

January 23, 2024

Contents

Chapter 1

Introduction

1.1 Motivation

Parser combinators [1] are an elegant approach for writing parsers in a manner which remains close to the original grammar specification. `parsley` [2] is a parser combinator library implemented as an embedded domain-specific language (DSL) [3] in Scala, with an API inspired by the `parsec` [4] family of libraries in Haskell. However, as with many libraries, there exists a learning curve to utilising `parsley` and parser combinator libraries in an idiomatic manner.

While well-documented, the wealth of information to get started with `parsley` can be overwhelming for users, particularly those new to parser combinators. Furthermore, there exists a number of design patterns [5] for writing maintainable parsers, which even experienced users may be unaware of. A potential solution to this problem is tooling to provide automated code hints, which a user can use during the development cycle to evaluate if their code adheres to best practices.

A number of modern integrated development environments (IDEs) provide code hints to warn programmers about problems in their source code, highlighting offending snippets and suggesting actions to improve suboptimal or incorrect code [6]. Many of these code analysis tools are designed to detect general issues for the host language, rather than specifically for libraries. However, tools may also utilise domain-specific code analyses in order to detect issues specific to a particular system or problem domain [7, 8, 9].

This project aims to explore the potential of harnessing code analysis techniques to develop a new tool, `parsley-garnish`, that offers code hints aimed at assisting programmers in writing idiomatic and correct `parsley` code. Additionally, for certain hints which can be automatically fixed, `parsley-garnish` will provide automated actions to resolve the issue. The goal of `parsley-garnish` is to be used as a companion library to `parsley`, in order to improve its ease of adoption and to help users enforce best practices.

1.2 Planned Objectives

Chapter 2

Background

2.1 Static Analysis Tools

Chapter 3

Project Plan

Chapter 4

Evaluation

Bibliography

[1] Hutton G. Higher-order functions for parsing. Journal of Functional Programming. 1992 Jul;2(3):323-43. Available from: https://doi.org/10.1017/S0956796800000411.

[2] Willis J, Wu N. Garnishing parsec with parsley. In: Proceedings of the 9th ACM SIGPLAN International Symposium on Scala. Scala 2018. New York, NY, USA: Association for Computing Machinery; 2018. p. 24-34. Available from: https://doi.org/10.1145/3241653.3241656.

[3] Hudak P. Building domain-specific embedded languages. ACM Comput Surv. 1996 dec;28(4es):196-es. Available from: https://doi.org/10.1145/242224.242477.

[4] Leijen D, Meijer E. Parsec: Direct Style Monadic Parser Combinators for the Real World; 2001. UU-CS-2001-27. User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 25-29, 2007. Available from: https://www.microsoft.com/en-us/research/publication/parsec-direct-style-monadic-parser-combinators-for-the-real-world/.

[5] Willis J, Wu N. Design patterns for parser combinators in scala. In: Proceedings of the Scala Symposium. Scala '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 9-21. Available from: https://doi.org/10.1145/3550198.3550427.

[6] Kurbatova Z, Golubev Y, Kovalenko V, Bryksin T. The IntelliJ Platform: A Framework for Building Plugins and Mining Software Data. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW); 2021. p. 14-7. Available from: https://doi.org/10.1109/asew52652.2021.00016.

[7] Renggli L, Ducasse S, Gîrba T, Nierstrasz O. Domain-Specific Program Checking. In: Vitek J, editor. TOOLS'10: Proceedings of the 48th International Conference on Objects, Models, Components, Patterns. vol. 6141 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2010. p. 213-32. Available from: https://doi.org/10.1007/978-3-642-13953-6_12.

[8] Gregor D, Schupp S. STLlint: lifting static checking from languages to libraries. Software: Practice and Experience. 2006;36(3):225-54. Available from: https://doi.org/10.1002/spe.683.

[9] xUnit net. xUnit.net Analyzers package;. Available from: https://github.com/xunit/xunit.analyzers.