

Chapter 1

Introduction

1.1 Motivation

Parser combinators [Hutton 1992] are an elegant approach for writing parsers in a manner that remains close to the original grammar specification. `parsley` [Willis and Wu 2018] is a parser combinator library implemented as an embedded domain-specific language (DSL) [Hudak 1996] in Scala, with an API inspired by the `parsec` [Leijen and Meijer 2001] 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 [Willis and Wu 2022] 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 [Kurbatova et al. 2021]. 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 [Renggli et al. 2010; Gregor and Schupp 2006].

1.2 Project Goals

This project aims to explore the potential of harnessing static 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 issues that 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.

Place this in the right bit (I think intro is good) As noted by Gibbons and Wu [2014], a deep-embedded DSL consists of two components:

- A representation of the language’s abstract syntax, in the form of the aforementioned datatype.
- Some traversals over the datatype, which gives *semantics* to that syntax.

A deep-embedded DSL and a linter for that DSL can share the same abstract syntax, but differ in the semantic interpretation of that syntax:

- The DSL semantics are evaluation. In this case, `parsley` interprets its syntax to output a parser.
- The linter’s semantics are pretty-printing. In this case, `parsley-garnish` interprets the syntax to output a human-readable representation of the parser.