Chapter 1

Project Plan

1.1 Proposed deliverables and milestones

This project is rather exploratory in nature, and as such, these milestones are highly subject to change as the project progresses. However, I have attempted to outline some key deliverables which I believe to be achievable, as well as some optional deliverables to explore if time permits.

1.1.1 Key deliverables

- Simple lint rules for non-idiomatic combinator usage which can be simplified with an automated refactoring. For example, `endBy(p, sep)` is the idiomatic equivalent to `many(p <* sep)`.
- Detection of left-recursive parsers, which are a bug pattern since they will cause infinite recursion at runtime. It should be possible to perform a static analysis to produce a code transformation to utilise the `postfix` family of combinators to eliminate left-recursion and resolve the bug.
- Lint rules to detect ambiguity in parsers, which can be a bug pattern since the parser may not behave as expected. It may be possible to provide an automated fix in some cases, but more research has to be done to determine the feasibility of this.

  Conversely, it may also be possible to detect overuse of the `atomic` combinator. Users may overuse this combinator to avoid ambiguity, but overuse is a code smell as it can negatively impact error messages reported by the parser.
- Proofs that all code transformations preserve the semantics of the original parser.

1.1.2 Optional deliverables

- Lint rules that warn against side-effecting parsers. `Parsley` employs aggressive parser optimisations which may assume that parsers are pure. The lint rule would remind users to use the `impure` combinator to avoid `parsley` from optimising away any intended behaviour.
- Lint rules to detect when a parser using `chain/infix/postfix` combinators can be rewritten to use a precedence table, and the automated refactoring to accompany this rule. Further lints/refactorings related to precedence tables may also be possible, such as looking at type information to determine the most appropriate shape (`Ops/SOps/GOps`).
- An editor plugin or extension to provide integration of `parsley-garnish` into the code editor. As explored in the background section, it is possible to provide this integration for Scalafix, but the difficulty of this task is unknown.
- Exploration of possible pedagogical applications of `parsley-garnish`. For example, lint rules may be useful for marking submissions for the Second Year WACC compilers project.

1.2 Current Progress

Most of the work so far has been on research and experimentation with Scalafix. Although there was good documentation to get started, Scalafix is an active project and the documentation is not always fully comprehensive or up-to-date. I still have some concerns about whether the Scalafix API is expressive enough to implement all of the analyses I have in mind, but I will continue to explore this as the project progresses.

From my experimentation, I have a relatively good idea of how to implement most of the "simple" lint rules.

At the time of writing, I have also implemented a proof-of-concept rule for transforming left-recursive parsers, on a small subset of `parsley`'s combinators. This has been encouraging but much work remains to be done to be able to apply the rule on more complex parsers. The current results of the transformations are also sometimes too complex, so I will need to investigate methods such as defunctionalisation and/or alternate methods of representing terms to simplify the output statically. Furthermore, the transformation is likely to be correct but not proven yet.

1.3 Project Timeline

My immediate plan is to continue work on the left-recursion transformation, while also working on the simpler lint rules. I hope to complete the left-recursion rule by early March, although unforseen difficulties may arise (such as if output simplification proves to be more difficult than expected). If it is not complete or close to complete by then, I will re-evaluate the feasibility of the rest of the milestones and adjust accordingly. I have already communicated to my supervisor that this left-recursion rule will likely be the most involved deliverable of the project, although there is also a chance that many of the techniques employed could be reused for other transformations.

The next two months will be spent exploring other milestones. I plan on approaching this by alternating research and implementation, as I have found that this has been a good way to make progress so far, while also guaranteeing that I will make progress on the final deliverable.

By the beginning of summer term, I would also like to have a very rough draft of the final report completed, based on the work completed so far. This will allow me to communicate with my supervisor on the state of the project, as well as leaving me ample time to edit and refine the report. The remaining milestones to cover will be decided during this meeting; at this point, having implemented a number of rules, I hope to have a much better grasp on how long each deliverable could take. This remaining time will then be divided between writing the report and implementing the agreed-upon milestones.