Incremental Type-Safe Structural Diffing

Jort van Gorkum

January 14, 2022

1 Introduction

- What is the problem? Illustrate with an example.
- What is/are your research questions/contributions?

2 Background

2.1 An Efficient Algorithm for Type-Safe Structural Diffing

The paper *An Efficient Algorithm for Type-Safe Structural Diffing* by Victor Cacciari Miraldo and Wouter Swierstra presents an efficient datatype-generic algorithm to compute the difference between two values of any algebraic datatype. In particular, the algorithm readily works over the abstract syntax tree (AST) of a programming language[2].

The algorithm when implemented in Haskell contains two main functions the `diff` and `apply`. The `diff` function computes the difference between two values of type `a`, and the `apply` function attempts to transform one value according to the information stored in the `Patch`.

```
diff  :: a -> a -> Patch a
apply :: Patch a -> a -> Maybe a
```

These functions are expected to fulfill some properties. The first being *correctness*: the patch that `diff x y` computes can be used to faithfully reproduces `y` from `x`.

```
∀ x y . apply(diff x y) x ≡ Just y
```

The second being *preciseness*:

```
∀ x y . apply(diff x x) y ≡ Just y
```

The last being *computationally efficient*: both the *diff* and *apply* functions needs to be space and time efficient.

The most commonly used diffing algorithm by version control systems is the UNIX `diff`[1] or also known as the Hunt-McIlroy algorithm. The UNIX `diff` satisfying these previously stated properties for `a ≡ [String]`[2]. Several attempts have been made to generalize this algorithm for arbitrary datatypes, but the way the UNIX `diff` represents the `Patch` using only *insertions, deletions* and *copies of lines* has two weaknesses. Firstly, the non-deterministic nature of the design makes the algorithm inefficient, and secondly, there exists no canonical 'best' patch and the choice is arbitrary[2].

Miraldo's and Swierstra's algorithm improves this shortcoming by introducing more operations: *arbitrary reordering, duplication* and *contraction of subtrees*. This restricts non-determinism, making it easier to compute patches and increasing the opportunities for copying.

3 Preliminary Results

- What examples can you handle already?
- What prototype have I built?
- How can I generalize these results? What problems have I identified or do I expect?

4 Timetable and Planning

- What will I do with the remainder of my thesis?
- Give an approximate estimation/timetable for what you will do and when you will be done.

A Appendix

References

[1] James Wayne Hunt and M Douglas MacIlroy. *An algorithm for differential file comparison.* Bell Laboratories Murray Hill, 1976.

[2] Victor Cacciari Miraldo and Wouter Swierstra. “An efficient algorithm for type-safe structural diffing”. In: *Proceedings of the ACM on Programming Languages* 3.ICFP (2019), pp. 1–29.