Incremental Type-Safe Structural Diffing

Jort van Gorkum

January 14, 2022

Todo list

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 Hunt-McIlroy algorithm used by the UNIX `diff` utility[1]. The UNIX `diff` satisfies 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.

Write a piece about how the diff function uses an oracle which uses cryptographic hashes

2.2 Sums of Products for Mutually Recursive Datatypes

The paper *Sums of Products for Mutually Recursive Datatypes* written by Victor Cacciari Miraldo and Alejandro Serrano presents a new approach to generic programming using recursive positions to handle mutually recursive families and the *sum-of-products* structure. This work (`generics-msrop`) is later used by the paper *An Efficient Algorithm for Type-Safe Structural Diffing* by Victor Cacciari Miraldo and Wouter Swierstra[2] to define the generic version of their diffing algorithm. Compared to existing generic programming libraries, `generics-mrsop` has *deep explicit recursion, sums of products* and supports *mutually recursive datatypes*.

Explicit recursion There are two ways to represent values. One contains the information on what properties of a datatype are recursive. The other does not contain that information. If we do not know explicitly if the property is recursive, then only one layer of the value can be formed into a generic representation. This is called *shallow* encoding. If we explicitly keep track of the recursive property, then the entire value can be transformed into a generic representation. This is called *deep* encoding. Using the *deep* encoding more datatypes can be defined generically (e.g., a generic *map* or generic Zipper datatype).

Sums of Products

Explain sums of products

Explain mutually recursive datatypes

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.