Setfield.jl: Changing the immutable

Jan Weidner[1]

[1]PTW-Dosimetry

ABSTRACT

We discuss the problem of updating immutable objects. The solutions presented are implemented in the Setfield.jl package [8].

Keywords

Julia, Functional Programming, Optics, Immutable

1. Overview

In the Julia programming language [1], some objects are **mutable** (Array, mutable struct, ...), while others are **immutable** (Tuple, struct, ...). Neither is strictly better than the other in every situation. However, immutability usually leads to code that is easier to reason about, for both humans and compilers. Which means less buggy and more performant programs. One convenience with mutability is, that it makes updating objects very simple:

```
spaceship.captain.name = "Julia"
```

The analogous operation in the immutable case is to create a copy of *spaceship*, with just the captains name changed to "Julia". This operation is sometimes called functional update. Just think for a moment, how would you do achieve this? It is a non trivial problem and there are many approaches, both in Julia [3] and other languages [2]; [6]; [5].
The Setfield.jl package [8] provides one solution to this problem. Namely it allows the user to specify a functional update using the same syntax as in a mutable setting. The only syntactic difference is the @set macro in front:

```
@set spaceship.captain.name = "Julia"
```

And voila, this returns an updated copy. The implementation is based on the lens formalism, see the documentation of the package. For an entry point to the lens literature see the introduction of [4] or references in [7].

2. Acknowledgements

I would like to acknowledge various larger and smaller contributions in form of issues and pull requests, by various authors. Especially Takafumi Arakaki made many great contributions. Details can be extracted from the github repository of the package.

3. References

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[2] Edward Kmett and contributers. Lens, 2012. https://github.com/ekmett/lens.

[3] Keno Fischer. Heap allocated immutable arrays and compiler support, 2019. https://github.com/JuliaLang/julia/pull/21912.

[4] Michael Johnson, Robert Rosebrugh, and Richard Wood. Algebras and update strategies. *Journal of Universal Computer Science*, 16(5):729–748, 2010. The following article appeared in Journal of universal computer science, 16(5), 729-748, and can be found at http://dx.doi.org/10.3217/jucs-016-05-0729. Version archived for private and non-commercial use with the permission of the author/s and according to publisher conditions. For further rights please contact the publisher.

[5] Lee Bryon and contributers. Immutable-js, 2014. https://github.com/immutable-js/immutable-js.

[6] Nathan Marz and contributers. Specter, 2015. https://github.com/redplanetlabs/specter.

[7] Mitchell Riley. Categories of optics. 2018.

[8] Takafumi Arakaki and Jan Weidner. Setfield.jl, 2017. https://github.com/jw3126/Setfield.jl.